VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Možnosti tvorby nových bloků pro systém Scilab/Xcos

Creation of New Blocks for Scilab/Xcos System

2019 Kamil Trubák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Kamil Trubák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Možnosti tvorby nových bloků pro systém Scilab/Xcos
Creation of New Blocks for Scilab/Xcos System

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat možnosti tvorby nových uživatelských bloků v simulačním prostředí Scilab/Xcos a následnou tvorbu uživatelských knihoven.

1. Popište stručně prostředí Scilab/Xcos.
2. Popište formát, ve kterém jsou jednotlivé funkční bloky v prostředí Xcos implementovány včetně stručné ukázky na již existujícím bloku.
3. Proveďte rešerši možností tvorby nových uživatelských bloků pro systém Xcos.
4. Vytvořte vlastní bloky pro systém Xcos dle pokynů vedoucího s detailním popisem jejich tvorby a začlenění do uživatelské palety bloků.
5. Popište možnosti sdílení uživatelsky vytvořených palet mezi jednotlivými uživateli.

Seznam doporučené odborné literatury:

[1] GOMEZ, Claude, Carey BUNKS, Jean-Philippe CHANCELIER, Francois DELEBECQUE, Maurice GOURSAT, Ramine NIKOUKHAH a Serge STEER, ed. Engineering and scientific computing with Scilab. New York: Springer Science+Business Media, 2013. ISBN 978-1-4612-7204-5.

[2] CAMPBELL, Stephen J, Jean-Philippe CHANCELIER a Ramine NIKOUKHAH. Modeling and simulation in Scilab/Scicos with Scicoslab 4.4. Second edition. New York: Springer, [2010]. ISBN 978-1-4419-5526-5.

[3] PÁLENÍKOVÁ, Kristýna. *Využití prostředků Scilab pro simulace zpracování signálů*. Ostrava, 2016. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava. Fakulta elektrotechniky a informatiky. Vedoucí práce Ing. Jan Skapa, Ph.D.

[4] PROCHÁZKA, Josef. *Využití prostředků Scicos pro simulace zpracování signálů [online]*. Ostrava, 2016 [cit. 2018-09-21]. Dostupné z: http://hdl.handle.net/10084/116232. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 26. dubna 2019 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2019 .....

Rád bych na tomto místě poděkoval svému vedoucímu bakalářské práce, Ing. Janu Skapovi, Ph.D., za ochotný přístup, všechny rady a připomínky.

Abstrakt

Cílem práce je popsat možnosti tvorby nových uživatelských bloků v simulačním prostředí Scilab/Xcos a následnou tvorbu uživatelských knihoven. Xcos je bezplatný balíček od vydavatele Scilabu, společnosti ESI Group, sloužící pro modelování a simulaci explicitních a implicitních dynamických systémů a je také jednoduchou náhradou za Simulink od společnosti MathWorks. Práce bude zahrnovat popis prostředí Scilab a Xcos a popis implementace bloku včetně ukázky na konkrétním bloku. Dále bude práce obsahovat popis vytvoření nového bloku a jeho následné vsazení do uživatelské palety bloků.

Klíčová slova: Scilab, Xcos, SciNotes

Abstract

The object of my work is describe possibilities of creation of new blocks in the simulation software Scilab/Xcos and subsequent cration of user library. Xcos is a free package from the Scilab publisher of ESI Group designed to modeling and simulating explicit and implicit dynamical systems and is also a simple replacement for Simulink from MathWorks. Work will include description of the Scilab and Xcos, and a description of the block implementation including a sample of particular block. In addition, my work will contain a description of the creation of a new block and its insertion into the user palette of blocks.

Key Words: Scilab, Xcos, SciNotes

Obsah

Seznam použitých zkratek a symbolů

MATLAB	–	MATrix LABoratory

Seznam obrázků

Seznam výpisů zdrojového kódu

Úvod

Tato práce je zaměřena na popis možností tvorby nových uživatelských bloků v open-source simulačním prostředí Scilab od společnosti ESI Group a jeho rozšířením Xcos a následnou tvorbu uživatelských knihoven, včetně rozborů bloků, ukázky implementace bloků a jeho začlenění do uživatelské palety bloků.

V první kapitole si řekneme něco o Scilabu a jeho nástavbě Xcos. Ukážeme si, jak se orientovat v prostředí Scilab, jak se dostat do prostředí Xcos a tato prostředí si popíšeme.

Ve druhé kapitole si detailněji popíšeme prostředí Xcos a ukážeme si základní operace s bloky. Popíšeme si také, jak jsou bloky v prostředí Xcos implementovány, rozebereme si zdrojový kód jednoho konkrétního bloku a ukážeme si také příklad zapojení s vybraným blokem.

Ve třetí kapitole si popíšeme a ukážeme různé metody vytváření nových bloků pro systém Xcos. Postupně budou ukázány 3 způsoby a to vytvoření nového bloku pomocí superbloku, dále vytvoření bloku pomocí bloku scifunc a v poslední řadě vytvoření blokové výpočetní funkce pomocí jazyka C.

Ve čtvrté kapitole vytvoříme vlastní blok pro systém Xcos s detailním popisem jeho tvorby a začlenění do uživatelské palety bloků.

V páté kapitole si popíšeme možnosti sdílení uživatelsky vytvořených palet mezi jednotlivými uživateli. Připomeneme si, jak vytvořit novou paletu, jak do ní vsadit blok, jak paletu exportovat do souboru a v poslední řadě jak ji jiný uživatel může použít.

1 Popis prostředí Scilab/Xcos

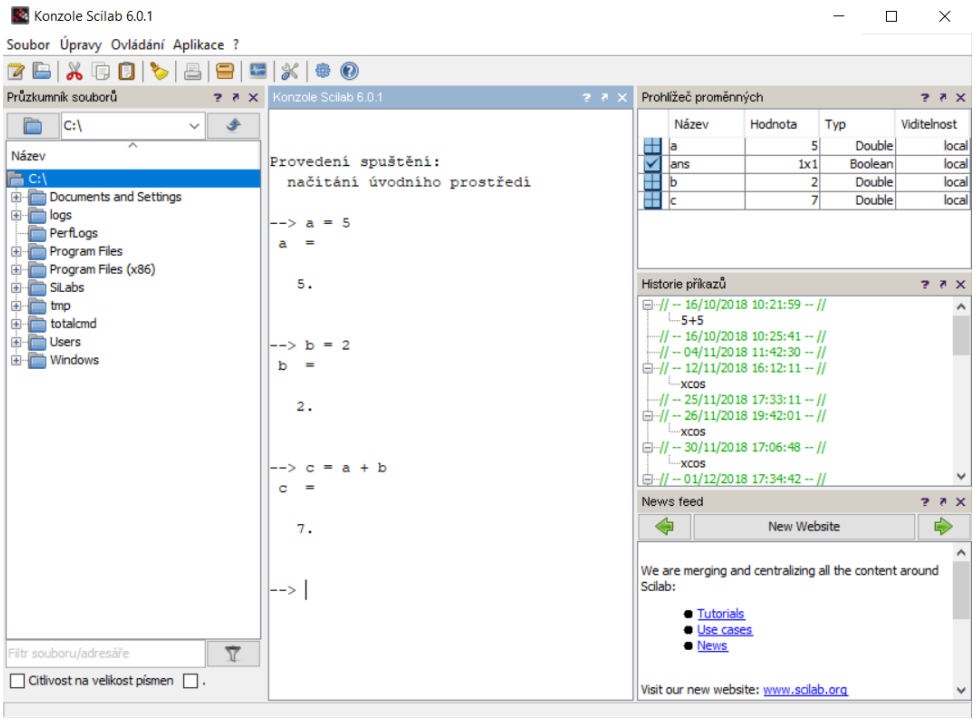
V této kapitole si popíšeme software Scilab, co to vlastně je, jak vypadá, pro co se dá využít a dále bych chtěl poukázat na Scilabovský bezplatný balíček Xcos.

1.1 Scilab

Scilab je volně šiřitelný program pro numerické výpočty podobný systému MATLAB. Program byl vytvořen francouzskými vědeckými institucemi INRIA a ENPC. Jeho licence umožňuje bezplatné používání. Od verze 5 je šířen pod svobodnou licencí CeCILL, kterou Free Software Foundation uvádí mezi licencemi kompatibilními s GPL. Nyní Scilab vyvíjí organizace ESI Group. Program umožňuje provádět jak jednoduchou aritmetiku, tak náročné výpočty. Scilab je například používán pro statistické analýzy, zpracování obrazu, simulace fyzikálních a chemických jevů. Zahrnuje také stovky matematických funkcí. Má dobrou úroveň programovacího jazyka, který umožňuje přístup k pokročilejším datovým strukturám, 2-D a 3-D grafickým funkcím. Dají se zde vytvořit programy, které lze rozdělit do dvou kategorií. Jednou z nich je procedura a druhá je funkce. Základní rozdíl těchto dvou variant je v tom, že pro funkci potřebujeme znát vstupní proměnné, ale pro proceduru nikoliv. Procedura se ukládá s příponou .sce a funkce s příponou .sci. Příklad funkce bude zobrazen a popsán níže v další kapitole.

Vzhledem k tomu, že syntaxe Scilabu je podobná MATLABu, Scilab obsahuje překladač zdrojového kódu pro pomoc při konverzi kódu z MATLABu na Scilab. Scilab je k dispozici zdarma pod licencí open-source. Vzhledem k otevřenému charakteru softwaru byly do hlavního programu začleněny některé příspěvky uživatelů.

Jak můžeme vidět z obrázku 1, prostředí Scilab není nijak zvlášť nepřehledné. Rád bych však popsal, co se v tomto prostředí nachází. Uprostřed můžeme vidět konzoli pro zadávání příkazů, spouštění různých balíčků a funkcí apod. Nalevo od konzole můžeme vidět průzkumník souborů, odkud také můžeme spouštět různé funkce, programy, atd. Vpravo nahoře se nachází prohlížeč proměnných, který nám zobrazuje všechny naše doposud vytvořené proměnné. Pod tím se pak nachází historie všech našich zadaných příkazů a úplně dole vlevo novinky a tutoriály Scilabu. [2, 3]



Obrázek 1: Pracovní plocha Scilab

1.2 Xcos

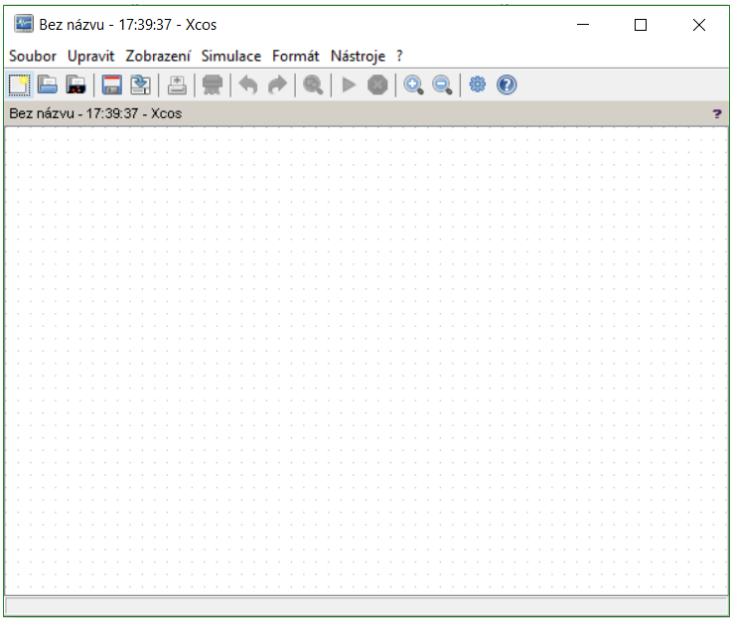
Scilab také zahrnuje bezplatný balíček nazvaný Xcos, který se snaží být jednoduchou náhradou za Simulink od společnosti MathWorks. Práce probíhá velmi podobně pomocí bloků. Tento balíček slouží pro modelování a simulaci explicitních a implicitních dynamických systémů, včetně jak souvislých, tak i diskrétních subsystémů, takže zde lze simulovat i složitější elektrické obvody, jednodušší děje v kapalinách, řešit diferenciální rovnice, modely slunečních soustav a mnoho dalšího. Ve složitých výpočtech bude poněkud těžkopádnější a pomalejší než Simulink. Jak prostředí Xcos vypadá a jak s ním pracovat si ukážeme níže v následující kapitole. [4]

2 Popis implementace bloků v prostředí Xcos.

V této kapitole si ukážeme, jak pracovat s prostředím Xcos, jak pracovat s jednotlivými bloky, jak zjistíme detaily o bloku, jakými způsoby jsou jednotlivé bloky v prostředí Xcos implementovány a ukážeme si také příklad na jednom konkrétním bloku.

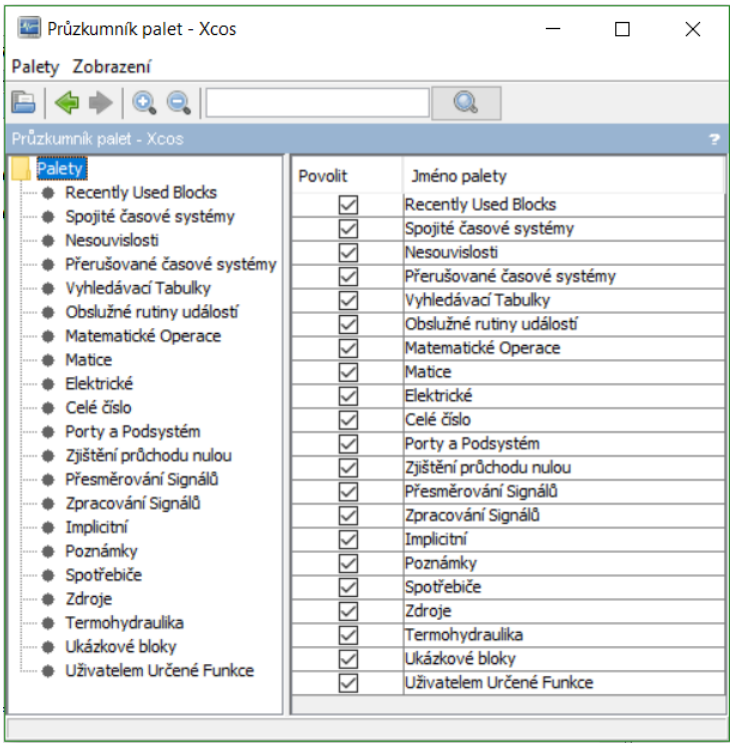
2.1 Prostředí Xcos

V prvé řadě si popíšeme, jak se dostaneme do prostředí Xcos. Do tohoto prostředí se můžeme dostat třemi způsoby. První z nich je přes konzoli ve Scilabu příkazem **xcos**. Po zadání tohoto příkazu se nám otevřou dvě okna. První z nich je pole pro vkládání bloků, které můžeme vidět na obrázku 2.



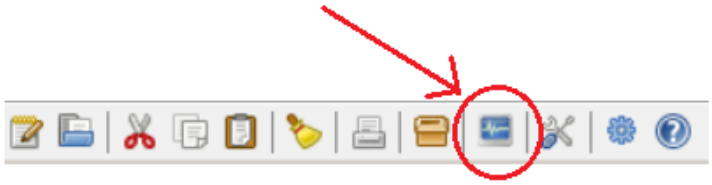
Obrázek 2: Pole pro vkládání bloků

Druhé okno, které se nám otevře, je samotný průzkumník palet obsahující konstrukční bloky, které můžeme vidět na obrázku3.



Obrázek 3: Průzkumník palet

Druhý způsob je přes nabídku v horní liště. Zde klikneme na nabídku **Aplikace** a v této nabídce klikneme na **Xcos**. Třetí Způsob je přes grafickou nabídku. Kde můžeme najít ikonu Xcos je zobrazeno na obrázku 4



Obrázek 4: Výběr Xcosu přes obrázkovou lištu

2.2 Popis palet a bloků v prostředí Xcos

V této podkapitole už si detailněji ukážeme průzkumník palet, popíšeme si, co můžeme vidět při rozbalení bloku a také jak jsou jednotlivé bloky implementovány.

Jakmile máme otevřený průzkumník palet, můžeme vidět všechny palety které nám Xcos nabízí od *Nejčastěji používaných bloků*, různé matematické, elektrické palety až po *Uživatelem Určené Funkce*. Když si rozklikneme náhodnou paletu, zobrazí se nám všechny bloky, které zvo-

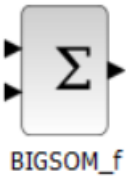
lená paleta nabízí. Když dvakrát klikneme na jednotlivý blok, přenese se nám do pole pro bloky. Jakmile opět dvakrát klikneme na blok v poli, objeví se nám tabulka pro nastavení parametrů bloku. Když na blok klikneme pravým tlačítkem myši, objeví se nám nabídka možností, co jde s blokem dělat. Pokud z nabídky zvolíme volbu **Nápověda bloku**, objeví se nám okno s informacemi o bloku. V tomto okně můžeme zjistit, jak je blok graficky zpracován, v jaké paletě se blok nachází, popis bloku, jaké parametry se v bloku nastavují, původní vlastnosti bloku, **Interfacing function** což je funkce napsaná v jazyce Scilab a slouží k definování vzhledu a chování grafických prvků bloku a **Computational function** neboli funkce, která je volána pro výpočet vlastností bloku a příklad zapojení se zvoleným blokem.

2.3 Implementace bloku v pracovním prostředí Xcos

Nyní už se dostáváme k podkapitole, ve které se podíváme na to, jak jsou jednotlivé bloky v prostředí Xcos implementovány. Ukážeme si, jak se k blokům dostat, jejich zdrojový kód, který si popíšeme a ukážeme si také další možnosti, jak s nimi pracovat.

V prvé řadě bych chtěl podotknout, že pro popis, rozbor a práci s bloky je také nutné vědět, kde máme Scilab nainstalován a kde jsou jednotlivé bloky uloženy. Cestu ke Scilabu si můžeme vypsat přímo v konzoli ve Scilabu příkazem **SCIHOME** (Příkaz musí být napsán velkými písmeny, jinak nefunguje). Pokud také chceme zjistit jaká lomítka používat, použijeme v konzoli příkaz **filesep**. Všechny bloky najdeme ve složce, do které jsme provedli instalaci programu Scilab a v podadresářích **...\scilab-6.0.1\modules\scicos_blocks\macros\...** Pokud se dostaneme až sem, máme na výběr z několika složek. Můžeme si v podstatě vybrat libovolnou z nich. Jakmile přejdeme do vybrané složky, objeví se nám několik souborů, buď s příponou .bin, nebo s příponou .sci. Jednotlivé soubory jsou vlastně bloky, které můžeme najít v průzkumníku palet v prostředí Xcos. Soubor s příponou .sci definuje blok jako **Interfacing function** , čili propojovací funkci, napsanou v jazyce Scilab, která definuje vzhled a chování grafických prvků bloku, jak už bylo zmíněno výše.

Pro ukázku kódu jsem si zvolil blok *BIGSOM_f*, který můžeme vidět na obrázku 5. Tento blok funguje jako sčítačka a můžeme ho nalézt v paletě *matematické operace*. [17]



Obrázek 5: Blok BIGSOM_f

Kód je napsán ve vestavěném scilabovském textovém editoru Scinotes. A nyní si po krocích rozebereme a popíšeme zdrojový kód.

```
function [x,y,typ]=BIGSOM_f(job,arg1,arg2)
    x=[];
    y=[];
     typ=[];
```

Výpis 1: Vstupní a výstupní hodnoty bloku BIGSOM_f

V tomto výpisu můžeme vidět název funkce, tedy *BIGSOM_f*, která bude použita uvnitř editoru SciNotes pro přístup k funkčnímu bloku. Dále můžeme vidět výstupní parametry **x**, **y**, **typ** a vstupní parametry **job**, **arg1**, **arg2**.

```
select job
```

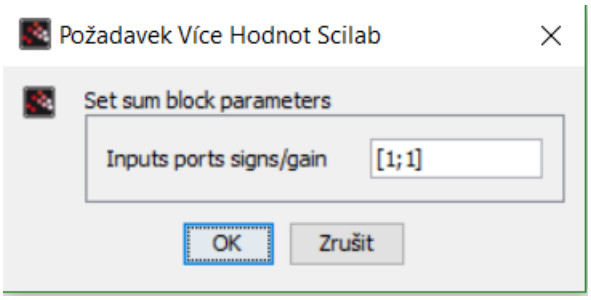
Výpis 2: 2. část programu

Tento příkaz slouží jako přepínač v jazyce C. V našem případě přepíná mezi možnostmi **set** a **define**.

Další část kódu definuje chování dialogového okna 6, které se objeví při dvojitém kliknutí na blok. Používá se zde funkce **getvalue()** k zobrazení dialogového okna a vrácení vložených hodnot. Všechny návratové hodnoty jsou uloženy ve struktuře proměnné **x**. **Graphics** je scilabovský objekt, který zahrnuje grafické informace vztahující se k vlastnostem bloku. **Model** je zase seznam, obsahující funkce bloku použité pro kompilaci bloku. **Exprs** nám pak udává řetězce formálních výrazů v dialogovém okně bloku.

```
        case "set" then
            x=arg1;
            graphics=arg1.graphics
            model=arg1.model
            exprs=graphics.exprs
            while %t do
                [ok,sgn,exprs]=scicos_getvalue("Set sum block parameters",..
                "Inputs ports signs/gain",list("vec",-1),exprs)
                if ~ok then
                    break
                end

                in = -ones(size(sgn,"*"),1)

                [model,graphics,ok] = check_io(model,graphics,in,-1,[],[]);
                if ok then
                    model.rpar = sgn(:);
                    graphics.exprs = exprs;

                    x.graphics = graphics;
                    x.model = model ;
                    break
                end

            end
```

Výpis 3: Chování dialogového okna bloku BIGSOM_f



Obrázek 6: Dialogové okno pro zadání parametrů bloku BIGSOM_f.

```
        case "define" then
                sgn=[1;1]
                model=scicos_model()
                model.sim=list("sum",2)
                model.in=[-1;-1]
                model.out=-1
                model.rpar=sgn
                model.blocktype="c"
                model.dep_ut=[%t %f]


                exprs=sci2exp(sgn)
                gr_i=[]
                x=standard_define([2 3],model, exprs,gr_i)
            end
        endfunction
```

Výpis 4: Vlastnosti modelu bloku BIGSOM_f

Tato část nám definuje vzhled funkčního bloku a inicializuje modelové proměnné, jakmile je blok poprvé umístěn do Xcos diagramu. Následující řádky, vyjmuty z kódu, nám definují vlastnosti modelu bloku. Příkaz **sci2exp** nám pak vrátí řetězec schopný generovat daný objekt Scilabu.

sgn=[1;1] - udává původně nastavené vstupní hodnoty
model=scicos_model() - definuje strukturu modelu
model.sim=list("sum",2) - odpovídá simulační funkci a funkčnímu typu.
model.in=[-1;-1] - odpovídá velikostem vstupních portů
model.out=-1 - odpovídá velikosti výstupního portu
model.rpar=sgn - odpovídá skutečnému vektoru parametrů
model.blocktype="c" - označení pro standartní blok
model.dep_ut=[%t %f] - odpovídá přímému průchodu a časové závislosti

Řádek **exprs=sci2exp(sgn)** definuje počáteční hodnoty, které budou zobrazeny v blokovém konfiguračním dialogu. Příkaz **sci2exp** pak převádí výraz **sgn** na řetězec. Následující řádek **gr_i=[]** slouží pro vykreslení blokových grafických a textových prvků. V našem případě se na blok nic nevpisuje. Poslední řádek **x=standard_define([2 3],model, exprs,gr_i)** definujte blok z jeho rozhraní jako v editoru Xcos.

Zde jsem čerpal ze zdroje [6, 10].

Jak si funkci uložit do bloku a do palety a jak z ní vytvořit binární soubor si ukážeme níže v podkapitole 2.6.

2.4 Ukázka a popis podrobností bloku vsazeného do pracovního prostředí Xcos

Nyní se můžeme na blok podívat přímo v pracovním prostředí Xcos. Po vsazení bloku do pracovního prostředí si můžeme zobrazit podrobnosti bloku. Ukážeme si jeho grafické a modelové vlastnosti a ve stručnosti si je popíšeme.

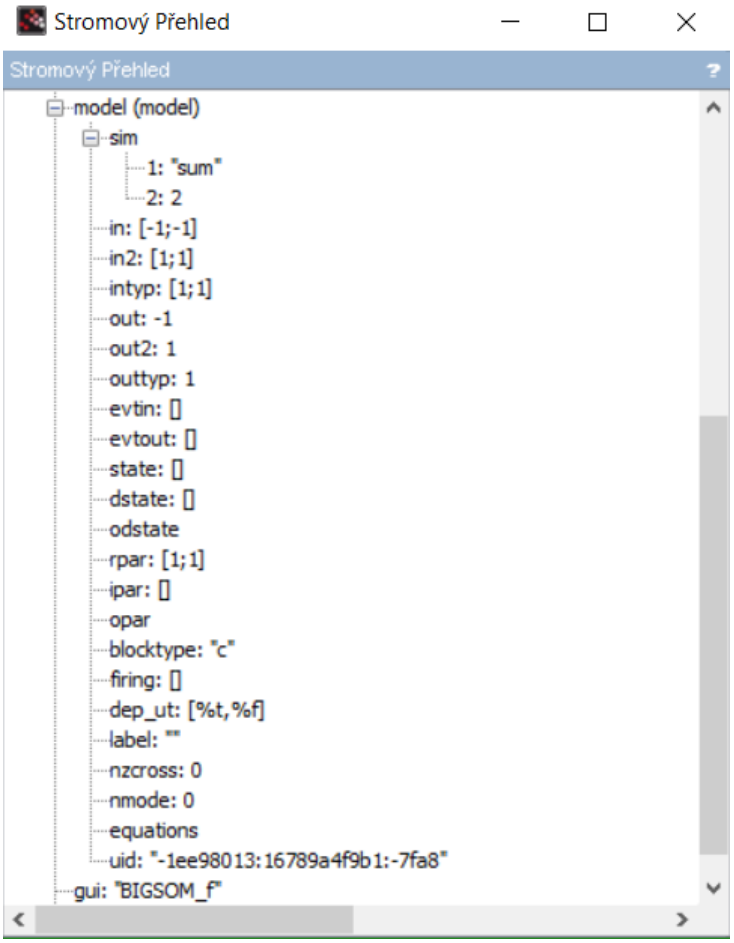
Obrázek 7: Grafické podrobnosti bloku BIGSOM_f

Na obrázku 7 můžeme vidět následující grafické vlastnosti bloku:

orig - udává, kde na ose x a y se blok nachází
sz - udává velikost bloku
exprs - udává hodnoty nastavené v dialogovém okně bloku.
pin - udává počet připojených linek na vstupy.
pout - udává počet linek připojených na výstup.
pein - vektor pein udává počet linek propojených k jejich událostním vstupním portům. Může být číslo 0, pokud událostní vstupní port není připojen .
peout - vektor peout udává počet linek propojených k jejich událostním výstupním portům. Může být číslo 0, pokud událostní výstupní port není připojen.
gr_i - znamená grafický výraz pro přizpůsobení grafického aspektu bloku.
id - identifikátor bloku.
in_implicit - udává, zda se jedná o explicitní nebo implicitní vstupní port. Tento vektor také udává chování každého vstupního portu.
out_implicit - udává, zda se jedná o explicitní nebo implicitní výstupní port. Tento vektor také udává chování každého výstupního portu.
in_style - vektor řetězců zahrnující grafické klíčové vlastnosti vstupních portů.
out_style - vektor řetězců zahrnující grafické klíčové vlastnosti výstupních portů.
in_label - vektor řetězců obsahující popisky vstupních portů.
out_label - vektor řetězců obsahující popisky výstupních portů.
style - název bloku.

Zde jsem čerpal ze zdroje [5].

Na dalším obrázku 8 můžeme vidět modelové vlastnosti bloku.



Obrázek 8: Modelové podrobnosti bloku BIGSOM_f

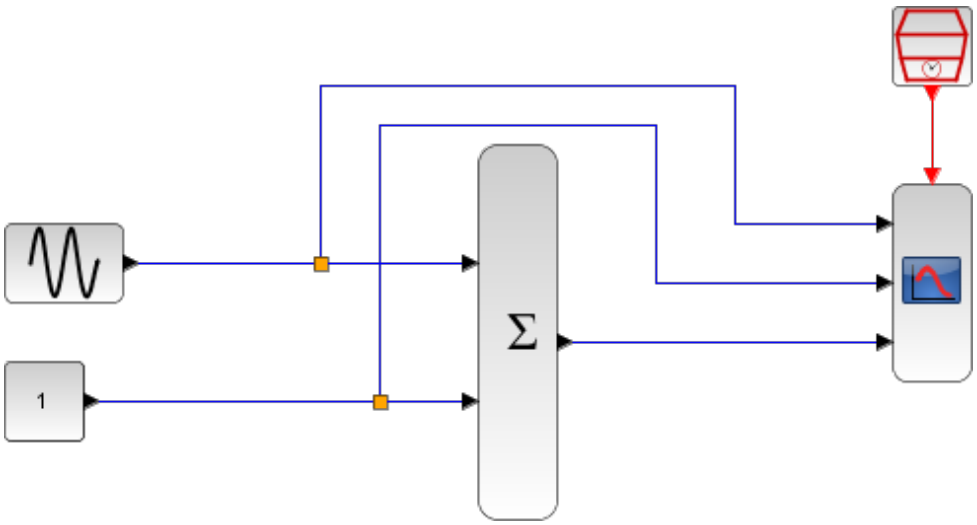
sim - obsahuje dvě složky. První udává název výpočetní funkce a druhá typ výpočetní funkce.
in - vektor udávající počet a velikost prvního rozměru regulérních vstupních portů indexovaných odshora dolů. Pokud je číslo záporné, pak se kompilátor pokusí najít takovou velikost, jaká by mu odpovídala. Pokud je číslo rovno nule, pak kompilátor ovlivní rozměr přidáním všech kladných velikostí nalezených ve vektoru. Pokud je číslo vyšší než nula, pak je velikost dána explicitně.
in2 - vektor, který specifikuje druhý rozměr regulérních vstupních portů indexovaných odshora dolů. Pokud je číslo záporné, pak se kompilátor pokusí najít takovou velikost, jaká by mu odpovídala. Pokud je číslo rovno nule, pak kompilátor ovlivní rozměr přidáním všech kladných velikostí nalezených ve vektoru. Pokud je číslo vyšší než nula, pak je velikost dána explicitně.
intyp - vektor udávající typ regulérních vstupních portů, kde jednička udává reálnou matici.
out - vektor udávající počet a velikost prvního rozměru regulérních výstupních portů indexovaných odshora dolů. Pokud je číslo záporné, pak se kompilátor pokusí najít takovou velikost, jaká

by mu odpovídala. Pokud je číslo rovno nule, pak kompilátor ovlivní rozměr přidáním všech kladných velikostí nalezených ve vektoru. Pokud je číslo vyšší než nula, pak je velikost dána explicitně.
out2 -vektor, který specifikuje druhý rozměr regulérních vstupních portů indexovaných odshora dolů. Pokud je číslo záporné, pak se kompilátor pokusí najít takovou velikost, jaká by mu odpovídala. Pokud je číslo rovno nule, pak kompilátor ovlivní rozměr přidáním všech kladných velikostí nalezených ve vektoru. Pokud je číslo vyšší než nula, pak je velikost dána explicitně.
outtyp -vektor udávající typ regulérních výstupních portů, kde jednička udává reálnou matici.
evtin - vektor udávající počet a velikost aktivačních vstupů.
evtout - vektor udávající počet a velikost aktivačních výstupů.
state - vektor obsahující počáteční hodnoty časově nepřetržitého stavu.
dstate - vektor obsahující počáteční hodnoty časově odděleného stavu.
odstate - vektor obsahující počáteční hodnoty objektového stavu.
rpar - určuje parametry s plovoucí desetinnou čárkou.
ipar - vektor celočíselných parametrů bloku.
opar - seznam parametrů objektů bloku.
blocktype - v našem případě znak c značí standartní blok.
firing - určuje počáteční dobu spuštění o velikosti odpovídající počtu aktivačních výstupních portů.
dep_ut - boolean vektor. %t znamená true, %f false. První vektor je true, pokud je blok vždy aktivní. Druhý vektor je true, pokud má blok přímý přívod, tj., alespoň jeden z výstupních portů závisí přímo na jednom ze vstupů. Jinak řečeno, když výpočetní funkce je volána příznakem 1, tak hodnota vstupu je použita pro výpočet výstupu.
label - definuje označení bloku.
nzcross - počet nulových přechodů.
nmode - délka režimu registru.
equations - používá se v případě implicitních bloků.

Zde jsem čerpal ze zdroje [6].

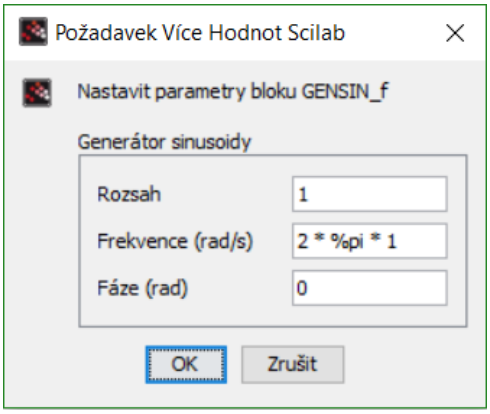
2.5 Příklad zapojení s blokem BIGSOM_f

V této podkapitole si ukážeme menší příklad zapojení s blokem *BIGSOM_f*. Půjde o součet jedničky se sinusovým signálem zobrazeným do grafického okna. V tomto zapojení použijeme generátor sinusového signálu *GENSIN_f* z palety *Zdroje* neboli *Sources palette*, konstantu 1 *CONST_m* z palety *Zdroje*, sčítačku *BIGSOM_f* z palety *Matematické operace* neboli *Math operations palette*, časový vzorkovač *SampleCLK* opět z palety *Zdroje* a zobrazovač signálů *CMSCOPE* z palety *Spotřebiče* neboli *Sinks*. Zapojení je zobrazeno na následujícím obrázku 9.



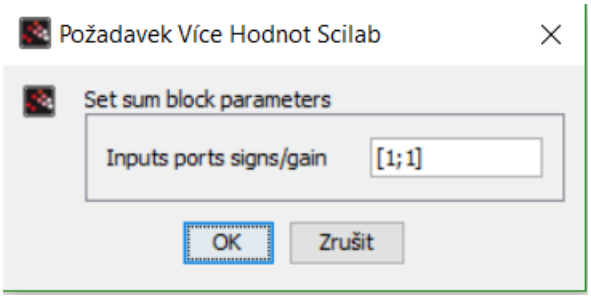
Obrázek 9: Zapojení pro součet signálu s konstantou

Pro správnost musíme nastavit náš sinusový signál. Z obrázku 10 lze vidět, jaké nastavení jsem volil. Rozsah je nastaven na jedničku, čili rozsah signálu bude od -1 do 1. Frekvence je pak **2***π. Fáze je rovna nule.



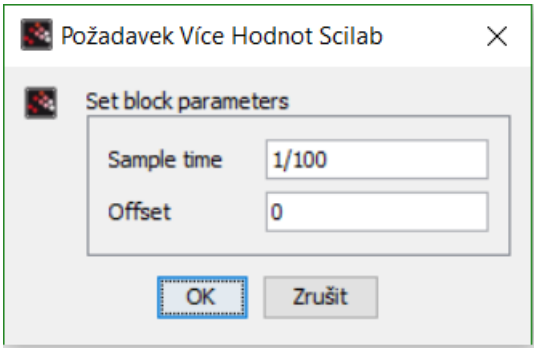
Obrázek 10: Nastavení hodnot generátoru sinusového signálu

Na sčítačce nastavíme vstupní znak a zisk na jedničku, jak jde vidět na obrázku 11.



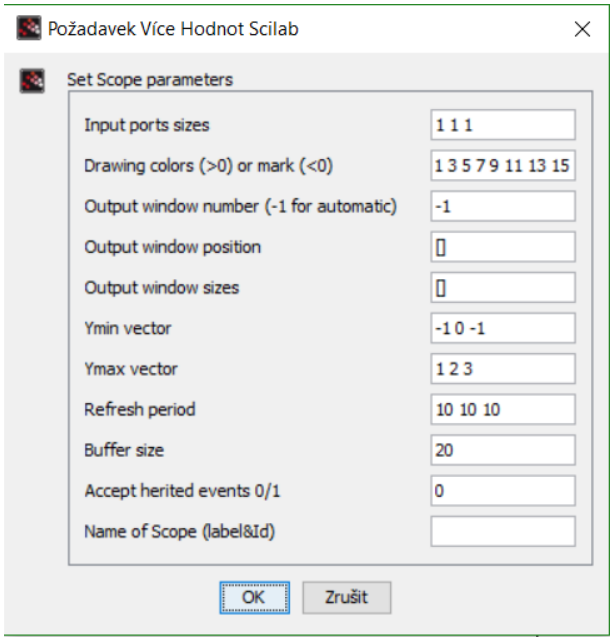
Obrázek 11: Nastavení parametrů sčítačky

Dále nastavíme časový vzorek na vzorkovači na 1/100 pro hezké vykreslení signálů. *Offset* nastavíme na nulu. Nastavení můžeme vidět na následujícím obrázku 12.



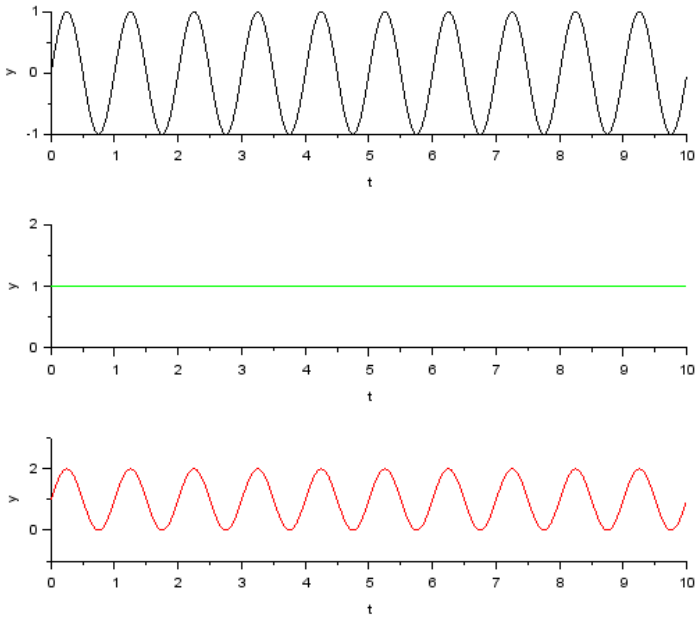
Obrázek 12: Nastavení vzorkování

V neposlední řadě také musíme nastavit vykreslovač signálů. Zde nastavíme počet vstupů na tři, velikosti os v kolonce *Ymin vector* a *Ymax vector* a periody signálů. Toto nastavení můžeme vidět na obrázku 13.



Obrázek 13: Nastavení vykreslení signálů

Jakmile máme nastaveno vše potřebné, můžeme zapojení spustit a vykreslit si naše signály. Z obrázku 14 můžeme vidět součet prvního a druhého signálu, který je vykreslen do třetího průběhu.



Obrázek 14: Vykreslení signálů

2.6 Ukázka zapojení s upraveným blokem

V této podkapitole si ukážeme, jak nově upravený blok uložit do nově vytvořené palety, jak z něj vytvořit binární soubor s příponou .bin a ukážeme si také ukázku zapojení s tímto blokem.

My si v našem bloku upravíme jen vstupní hodnoty **sgn** z 1 a 1 na -1 a -1. Čili sgn=[-1;-1]. První -1 nám udává že z přivedeného signálu se stane signál opačný a druhá -1 udává to, že se druhý signál bude od původního odečítat. Čili ze sčítačky se stane odečítačka. Po upravení bloku musíme blok vsadit do jedné z vytvořených palet a také musíme bloku přiřadit nějaký obrázek. Jak lze vidět na následujícím výpisu, vytvoříme si novou paletu s názvem **My palette** a přiřadíme do ní blok. Jakmile tak učiníme, nemusíme Scilab restartovat, ale můžeme rovnou přejít do prostředí Xcos a ihned začít pracovat s nově vytvořeným blokem. My si zde vytvoříme blok *BIGSOM_f_example*. Nesmíme také hlavně zapomenout naši funkci přejmenovat a spustit, aby se nám vytvořil nový soubor s příponou .sci. Ukázku kódu s vytvořením nové palety a začlenění bloku do palety můžeme vidět na výpisu 5.

```
loadXcoss();
o =  BIGSOM_f_example("define");
pal = xcosPal("My palette");
pal = xcosPalAddBlock(pal, o, SCI + '\modules\xcos\images\palettes\
    BIGSOM_f.png', SCI + '\modules\xcos\images\blocks\SUM.svg')
xcosPalAdd(pal);
```

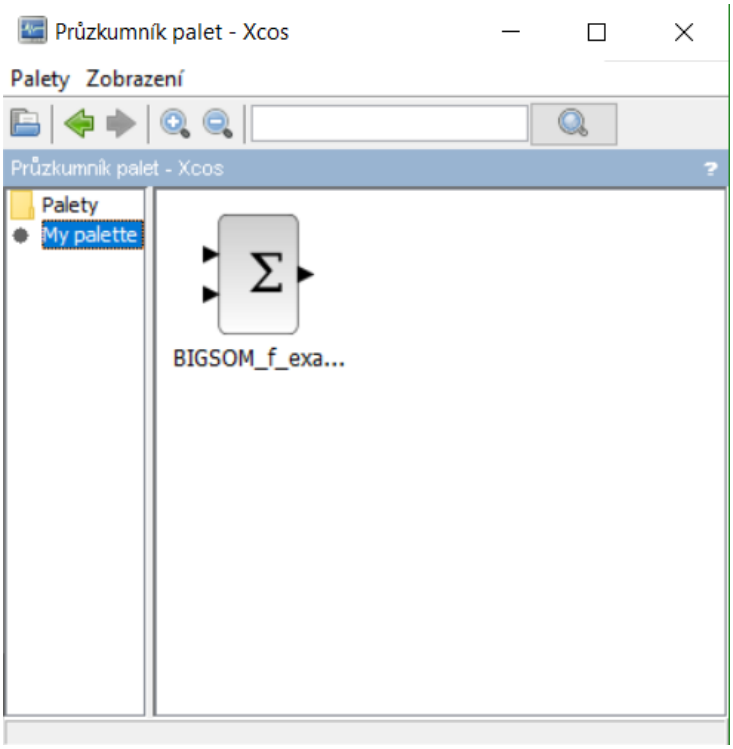
Výpis 5: Uložení bloku do palety My palette

Zde jsem čerpal ze zdroje [8, 9].

Binární soubor s příponou .bin z naší scilabovské funkce *BIGSOM_f.sci* potom vytvoříme pomocí příkazu **save**. Musíme zadat cestu, kam se má soubor vytvořit a také název naší funkce. Takto tento příkaz funguje jen na verzi 5.4.X a 6 a výše! Pro nižší verze Scilabu musíme vymazat uvozovky ve druhém vstupním argumentu. [7]

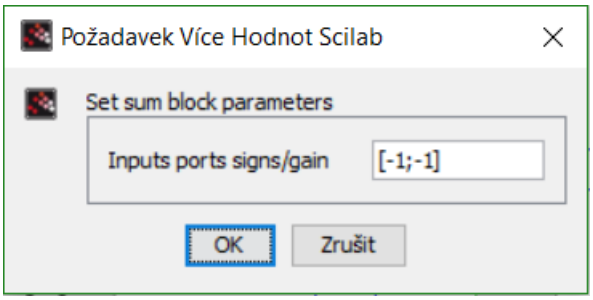
```
save(SCI + '\modules\scicos_blocks\macros\Linear\BIGSOM_f_example.bin'
    , 'BIGSOM_f_example');
```

Výpis 6: Vytvoření souboru BIGSOM_f_example.bin



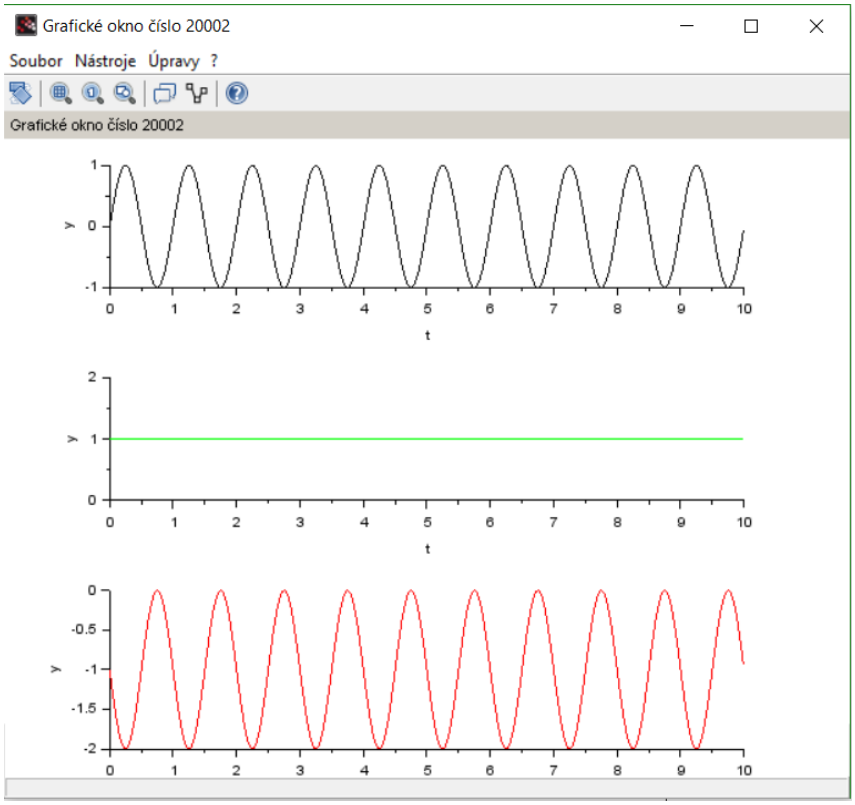
Obrázek 15: Nově vytvořená paleta s blokem BIGSOM_f_example

Nyní si ukážeme příklad zapojení s upraveným blokem. Příklad zapojení můžeme vidět na obrázku 9. Místo bloku *BIGSOM_f* si ale vsadíme náš nově upravený blok *BIGSOM_f_example* z palety *My palette*, který lze vidět na obrázku 15. Když 2x klikneme na náš blok, můžeme vidět změněné vstupní hodnoty. Obrázek 16.



Obrázek 16: Vstupní hodnoty bloku

Všechno nastavení ponecháme tak, jak máme v původním příkladu zapojení v podkapitole 3.5 a zapojení spustíme. Nyní se nám vykreslí tři signály. Sinusový signál s amplitudou 1, jednička signál a rozdíl těchto dvou signálů, jak lze vidět na obrázku 17.



Obrázek 17: Vykreselní signálů

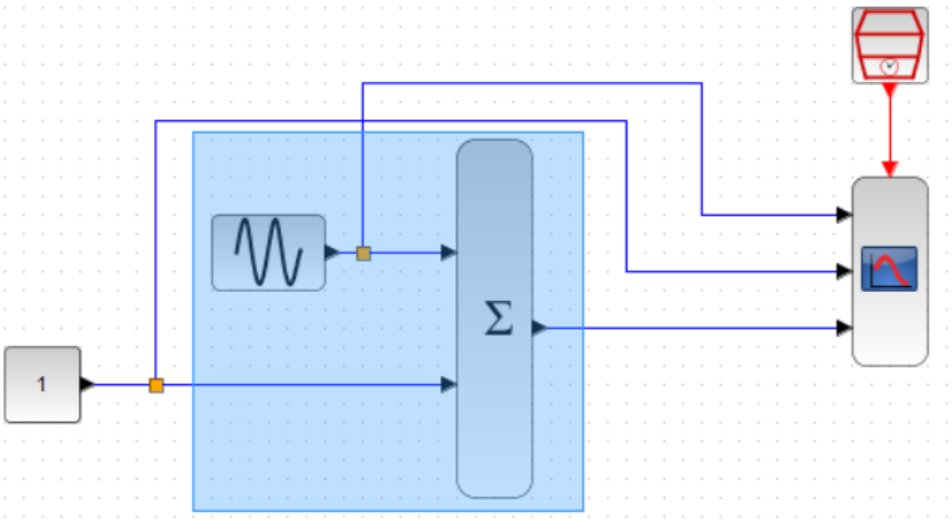
3 Možnosti tvorby nových uživatelských bloků pro systém Xcos

V této kapitole si popíšeme, jakými způsoby se dají vytvořit nové uživatelské bloky pro systém Xcos.

3.1 Vytvoření bloku jako superblok

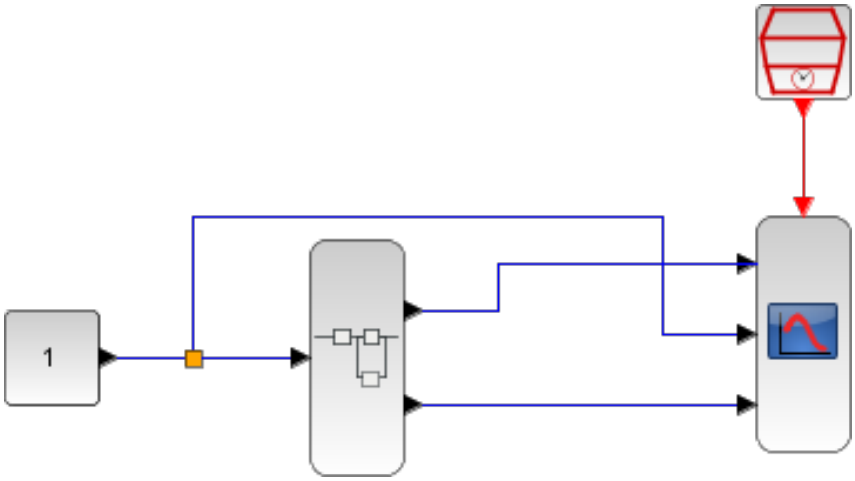
V této podkapitole si popíšeme a ukážeme, jak vytvořit nový uživatelský blok jako takzvaný *superblok*. Superblok nám slouží pro editaci nového blokového diagramu, který bude sestaven uvnitř superbloku. Funkci superbloku bude udávat funkce jeho vnitřního diagramu. Pro vytvoření takzvaného superbloku budeme pracovat v prostředí Xcos. Opět máme více možností, jak superblok vytvořit.

První z těchto možností je si nejdříve sestavit libovolný diagram a poté ho převést na superblok. Pro ukázku použijeme již dřívější známý příklad zapojení s blokem *BIGSOM_f*. My si v tomto zapojení převedeme do superbloku generátor sinusového průběhu a sčítačku. Konstantu 1 si necháme normálně mimo superblok, která bude připojena na jeho vstup. Zobrazovač signálů a časový vzorkovač taktéž nebudeme převádět, abychom měli na našem superbloku nějaké výstupy. Převedení docílíme tím, že si označíme bloky které chceme převést a popřípadě i uzly. Ukázku výběru můžete vidět na následujícím obrázku 18.



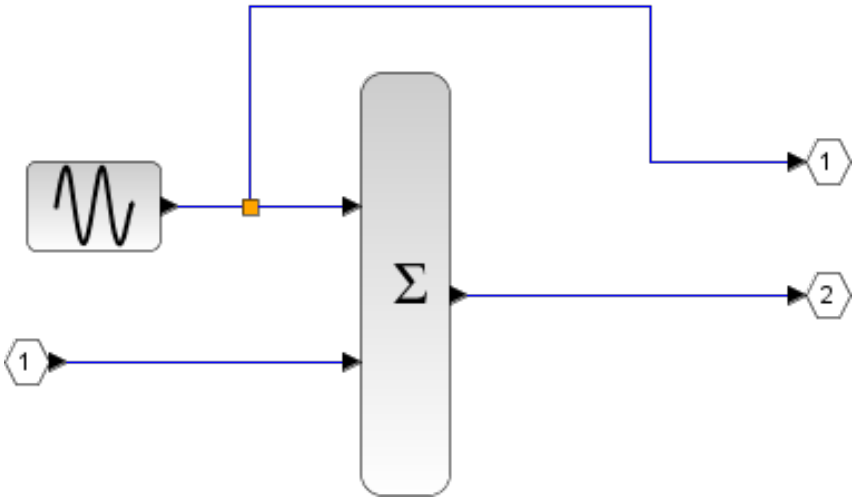
Obrázek 18: Výběr bloků pro superblok

Jakmile máme vybráno, najdeme v liště pracovního prostředí volbu upravit a zde vybereme volbu **Selection to superblock**. Nyní se nám naše tři bloky převedly do jednoho superbloku, jak lze vidět na následujícím obrázku 19.



Obrázek 19: Zapojení se superblokem

Po rozkliknutí se nám otevře okno s diagramem uvnitř superbloku (obrázek 20), ve kterém můžeme zapojení libovolně upravovat. Jak lze vidět, krajní bloky s čísly 1 a 2 napravo jsou výstupy našeho superbloku a blok s jedničkou nalevo je pak vstup našeho superbloku.

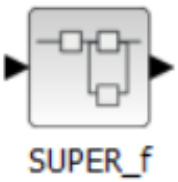


Obrázek 20: Uvnitř superbloku

Náš superblok si také můžeme pojmenovat a to buď klávesovou zkratkou Ctrl+F2, nebo pravým kliknutím na superblok, vybereme **formát** a **upravit**. Pokud máme v zapojení nějaké neznámé, bude nejlepší si vytvořit masku superbloku, ve které těmto neznámým přiřadíme výchozí hodnoty. Po tomto kroku, kdykoliv když poklikáme na superblok, se objeví tabulka, do které si navolíme potřebné hodnoty.

Pokud se stane, že simulace zapojení nelze spustit, je třeba rozkliknout superblok a znovu potvrdit parametry v jednotlivých blocích. Poté se může na spojích objevit varovná hláška, která nám říká, že jsou porty již připojeny, ať spoje propojíme jinam. Zde stačí spoje pouze odstranit a znova propojit. Poté by mělo spuštění simulace fungovat bez problémů.

Další možností je pracovat přímo s blokem *superblok*. Tento blok najdeme v paletě *User defined functions palette* neboli *Uživatelem určené funkce*. Zde najdeme náš potřebný blok jako *SUPER_f*, zobrazený na následujícím obrázku [21].

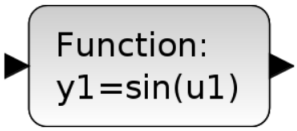


Obrázek 21: Ikona superbloku

Po vsazení bloku do pracovního prostředí si blok rozklikneme. Po rozkliknutí se nám otevře nové pracovní prostředí, ve kterém můžeme vidět pouze dva bloky a to vstupní a výstupní port. Nyní můžeme mezi tyto dva bloky libovolně vkládat bloky a vytvořit si své vlastní zapojení. Po vytvoření stačí zavřít a můžeme potom dále se superblokem pracovat podle potřeb. [16]

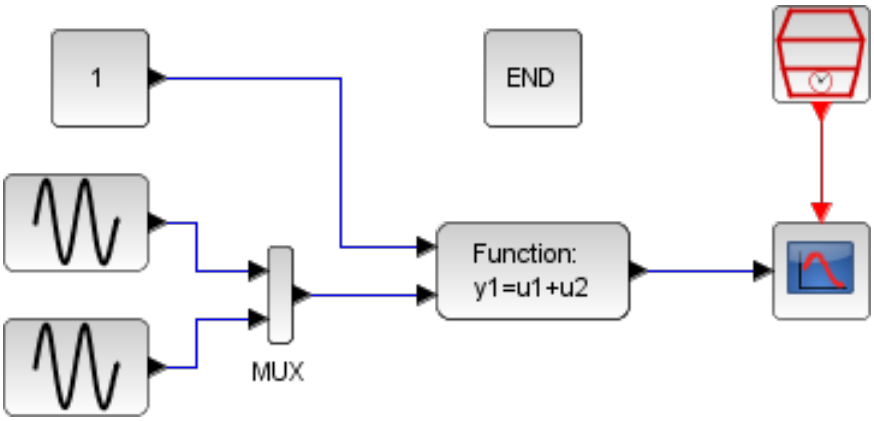
3.2 Vytváření bloků pomoci funkce scifunc

Další možností, jak vytvořit nový blok, je vytvořit ho pomocí scilabovského funkčního bloku *scifunc_block_m*, který je na následujícím obrázku 22. Najít ho můžeme v paletě *Uživatelem Určené Funkce* neboli *User defined functions palette*.



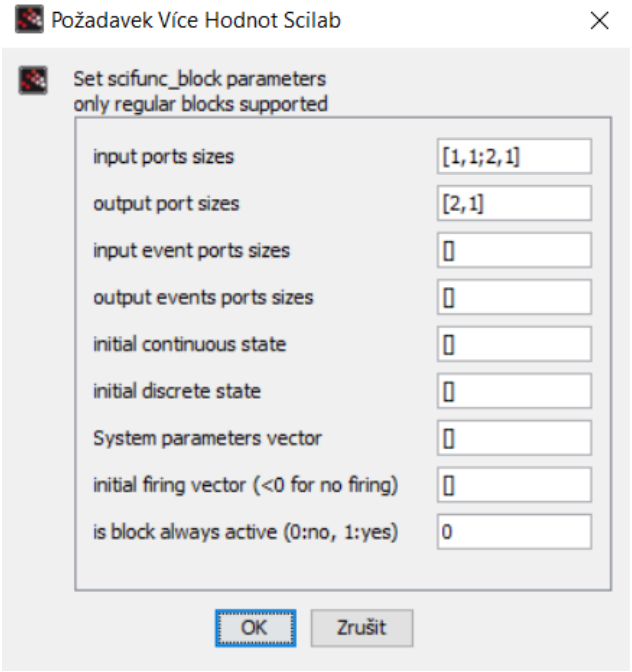
Obrázek 22: Ikona scifunc bloku

Tento blok dokáže simulovat jakýkoliv jiný blok. Jeho realizace je však už poněkud složitější. Funkce bloku se definují interaktivně pomocí dialogových oken a scilabovského jazyka. Čili uživatel si sám určuje počet vstupních portů, počet výstupních portů, počet vstupních událostních portů, počet výstupních událostních portů, počáteční kontinuální stav, počáteční diskrétní stav, systémové parametry vektoru, počáteční spouštěcí vektor a pokud je blok neustále aktivní. Po tomto nastavení se nám postupně objevují další dialogová okna. Není však nutné všechna okna vyplňovat. Vždy podle potřeby. Pro lepší ukázku si ukážeme příklad, jak se dá *scifunc* blok využít. Příklad můžeme vidět na obrázku 23. [15]



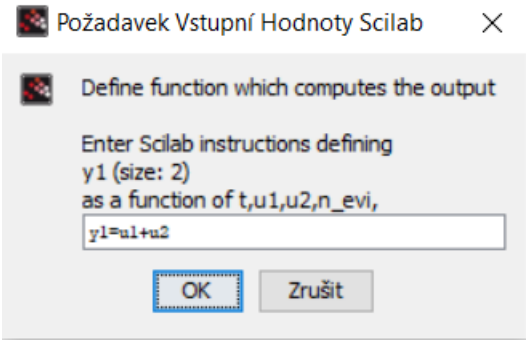
Obrázek 23: Příklad zapojení s blokem scifunc

Při poklikání na blok se nám otevře již výše zmíněné první dialogové okno, které je na dalším obrázku 24, do kterého se zadají blokové parametry. Jak je zde vidět, velikosti vstupních portů se zadávají ve formě matic. V našem případě je první vstup *u1* o velikosti 1x1, jelikož nám zde vchází pouze konstanta 1. Druhý vstup *u2* už je pak velikosti 2x1, protože nám zde vcházejí dva signály v jednom multiplexu. Výstup je pak také velikosti 2x1.



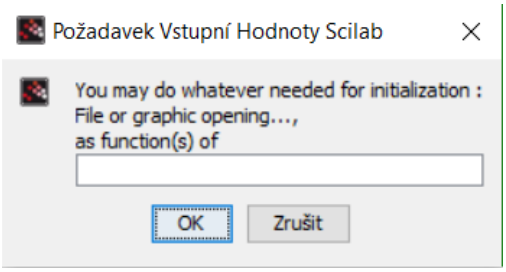
Obrázek 24: 1. dialogové okno scifunc

Jakmile se zadají a potvrdí, otevře se nám nové dialogové okno. V dalším okně 25 definujeme výpočetní funkci bloku. Definujeme si zde výstupní funkci *y1*. Parametry *u1* a *u2* jsou potom, jak již bylo zmíněno výše, hodnoty na vstupních portech. Parametr *t* je pak aktuální čas simulace.



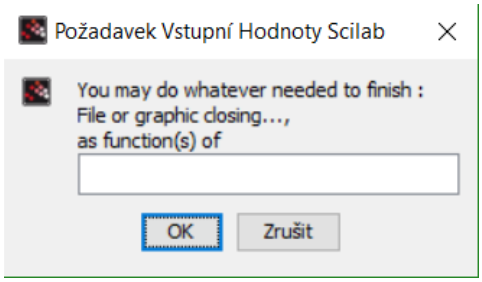
Obrázek 25: 2. dialogové okno scifunc

Při potvrzení tlačítkem ok se nám opět otevře další dialogové okno, jak lze vidět na následujícím obrázku 26, ve kterém můžeme udělat cokoliv chceme pro inicializaci.



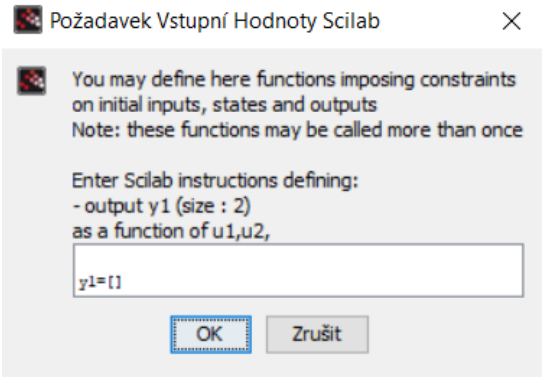
Obrázek 26: 3. dialogové okno scifunc

Další dialogové okno 27 je podobné oknu předešlému, jen se místo funkcí pro inicializaci zadávají funkce pro ukončení.



Obrázek 27: 4. dialogové okno scifunc

Poslední dialogové okno 28 nám umožňuje definovat funkce, které ukládají nějaká omezení na počáteční vstupy, stavy a výstupy.



Obrázek 28: 5. dialogové okno scifunc

Po všech těchto úkonech máme blok připravený k dalšímu využití.
Hlavní nevýhodou tohoto bloku ale je, že si nemůžeme podle sebe upravit, jak budou dialogová okna vypadat.

3.3 Vytváření bloků pomocí programovacího jazyka C

V této podkapitole si ukážeme jak se vytváří bloky pomocí programovacího jazyka C. Pomocí jazyka C se vytvářejí výpočetní funkce bloku. Výpočetní funkce je volána simulátorem různými způsoby. Způsob, jak je blok nazýván, je charakterizován typem rozhraní.

```
#include "textit_block.h"
#include <math.h>

void my_block(textit_block *block, int flag)
{
   ...
}
```

Výpis 7: Argumenty bloku v C

Flag, neboli příznak, indikuje úlohu, kterou musí výpočetní funkce vykonávat. To obvykle spočívá v aktualizaci některých polí blokové struktury. Úlohy mohou být následující:

flag	funkce	popis
0	volání integrátoru	vypočte derivaci spojitého časového stavu
1	výpočet výstupů	vypočítá výstupy bloku
2	aktualizace stavu	aktualizuje stavy buď vzhledem k externí aktivaci, nebo vzhledem k internímu nulovému přechodu
3	plánovač událostí	zpoždění aktivace výstupu
4	inicializace	inicializace stavů a dalších inicializací
5	ukončení	volá se před ukončením nebo při chybě simulace
6	inicializace, výpočet fixních bodů	pro omezení, která musí být splněna v počáteční době
7	vlastnosti spojitých stavových proměnných	nastavení vlastností spojitých stavových proměnných
9	úrovně a nulové přechody	pro vyhodnocení nulových přechodů a nastavení režimů, block->mode
10	výpočet jakobiánu	vypočte jakobián matice

Nyní si uvedeme příklad výpočetní funkce. Jako příklad použijeme kód sčítačky v C:

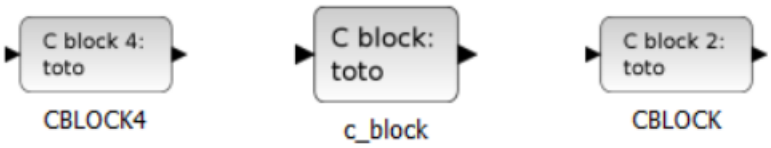
```
#include "textit_block.h"
#include <math.h>

void summation(textit_block *block, int flag)
{
   int j,k;
   if(flag == 1) {
      if(block->nin == 1) {
         block->outptr[0][0] = 0.0;
         for(j=0; j<block->insz[0]; j++) {
            block->outptr[0][0]=block->outptr[0][0]+block->inptr[0][j];
         }
      }
   else {
      for(j=0; j<block->insz[0]; j++) {
         block->outptr[0][j]=0.0;
         for(k=0; k< block->nin; k++) {
            if(block->ipar[k]>0) {
               block->outptr[0][j]=block->outptr[0][j]+block->inptr[k][j];
               }else {
                block->outptr[0][j]=block->outptr[0][j]-block->inptr[k][j];
               }
            }
         }
      }
   }
}
```

Výpis 8: Výpočetní funkce sčítačky

Následují vysvětlivky jednotlivých struktur.
block-> je vždy vektor na nějakou strukturu.
int nin - počet vstupů
double **outptr - tabulka ukazatelů na výstupy
int *insz - vstupní velikosti
double **inptr - tabulka ukazatelů na vstupy
int *ipar - celočíselné (integerovské) parametry velikosti nipar

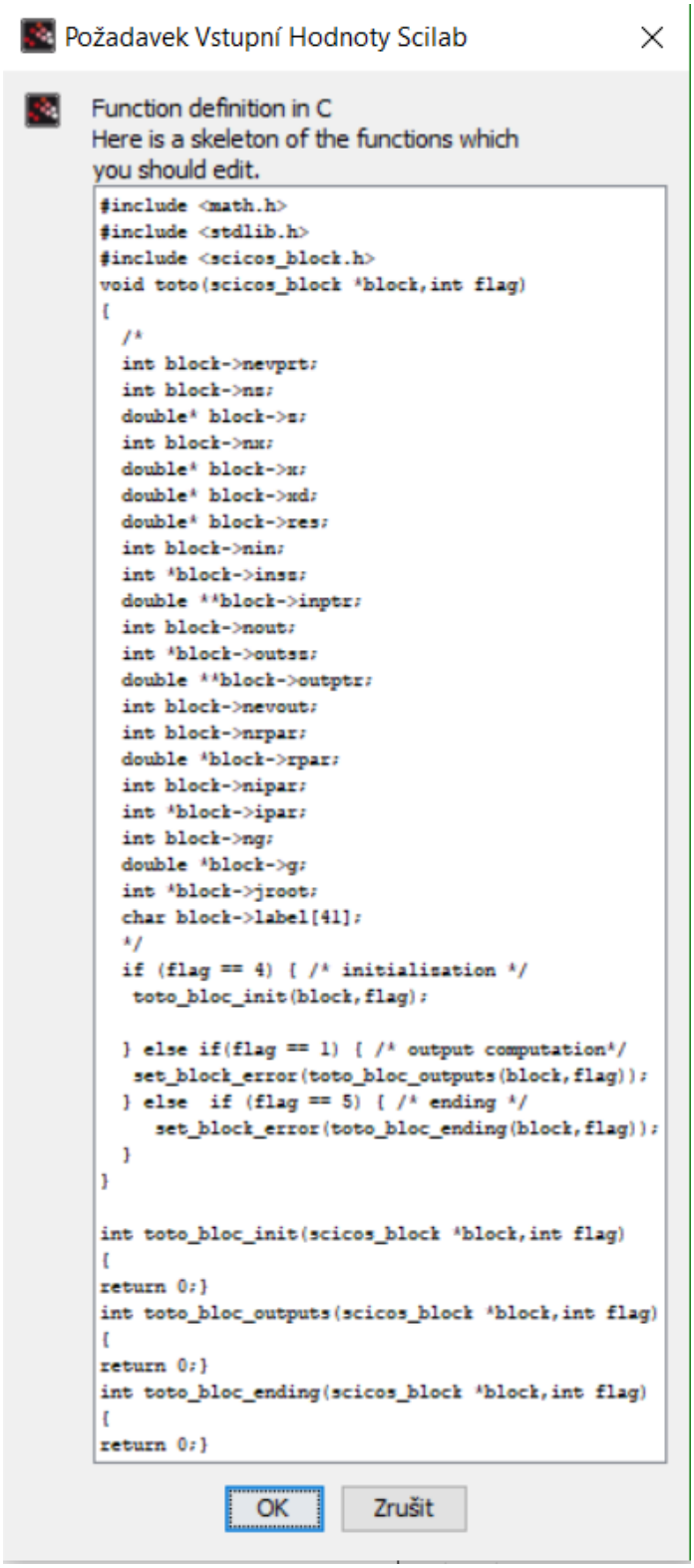
int nipar - počet celočíselných parametrů

Vytváření funkcí v C nám dokonce umožňuje i několik bloků v Xcos. Tyto bloky se od sebe liší hlavně obtížností. blok *c_block* je nejjednodušší, další bloky již vyžadují pro práci více parametrů. Najít je můžeme v paletě *Uživatelem Určené Funkce* neboli *User defined functions palette.* Vidět je můžeme na následujícím obrázku 29.



Obrázek 29: Bloky pro vytvoření funkce v C

Pro příklad si ukážeme blok *CBLOCK*. Po rozkliknutí bloku se nám otevře dialogové okno pro pojmenování funkce a pro nastavení dalších blokových parametrů. Nechal jsem zde výchozí název funkce *toto*. Jakmile potvrdíme, objeví se nové dialogové okno (na obrázku 30), ve kterém můžeme vidět knihovny, dále void funkci pro napsání výpočetní funkce, funkci *init* pro nastavení inicializace, funkci *outputs* pro nastavení výstupů a v poslední řadě funkci *ending* pro ukončení fungování bloku.



Obrázek 30: Okno pro vytvoření výpočetní funkce

V této podkapitole jsem čerpal ze zdrojů [1, 12, 18] .

4 Vytvoření bloku pro systém Xcos a jeho začlenění do palety uživatelských funkcí

V této kapitole si vytvoříme blok pro systém Xcos. Bude se jednat o *buffer*. Popíšeme si všechny kroky při tvorbě a následně tento blok začleníme do nově vytvořené palety.

V prvé řadě bychom si měli ujasnit, co to buffer je a jaká je jeho funkce. Buffer je v překladu takzvaná vyrovnávací paměť. Jedná se o paměť, do které jsou přivedena a následně střádána data těsně před jejich přesunem někam jinam. Buffer lze implementovat jako frontu FIFO (First In First Out). Čili data, která přijdou jako první, tak také jako první odejdou. Do bufferu jsou například ukládána data z klávesnice, takže se znaky budou sázet v takovém pořadí, v jakém je napíšeme. Buffer také usnadňuje komunikaci mezi zařízeními, které mají rozdílnou rychlost. V našem případě bude buffer vysílat signály v takovém pořadí, v jakém do něj přijdou s uživatelem danou rychlostí. Uživatel také bude mít možnost si zvolit velikost bufferu neboli kolik signálů bude vysílat. [14]

4.1 Vytvoření bloku

Pro vytvoření bloku jsem se rozhodl pracovat s blokem *CBLOCK*. Tento blok si vložíme do pracovního prostředí Xcos a následně roklikneme. Otevře se nám první dialogové okno s nabídkou úprav vstupů, výstupů, atd. Jediné co si zde zatím upravíme je název funkce. Ten z *toto* přepíšeme na *buffer*. Tato změna však není nutná a poslouží nám pouze pro lepší přehlednost. Potvrdíme tlačítkem **ok**. Nyní se dostáváme do okna pro definici C funkce našeho bloku, které jsme již viděli v předešlé kapitole na obrázku 30. Nás však zajímá pouze výstup bloku, přesněji funkce *buffer_bloc_outputs*. V prvé řadě velikosti výstupu přiřadíme proměnnou *n*, se kterou také budeme později pracovat. Dále si zavedeme proměnnou *i*, kterou budeme potřebovat v následujícím cyklu. Vytvoříme si tedy cyklus, který bude zpracovávat signály až do velikosti výstupu. Výpis kódu lze vidět níže.

```
int buffer_bloc_outputs(textit_block *block,int flag)
{
   int n=block->outsz[0];
   int i;
   for(i=0;i<n-1;i++) {
       block->outptr[0][n-i-1]=block->outptr[0][n-i-2]; }
   block->outptr[0][0]=block->inptr[0][0];
   return 0;
}
```

Výpis 9: Výstupní funkce bloku

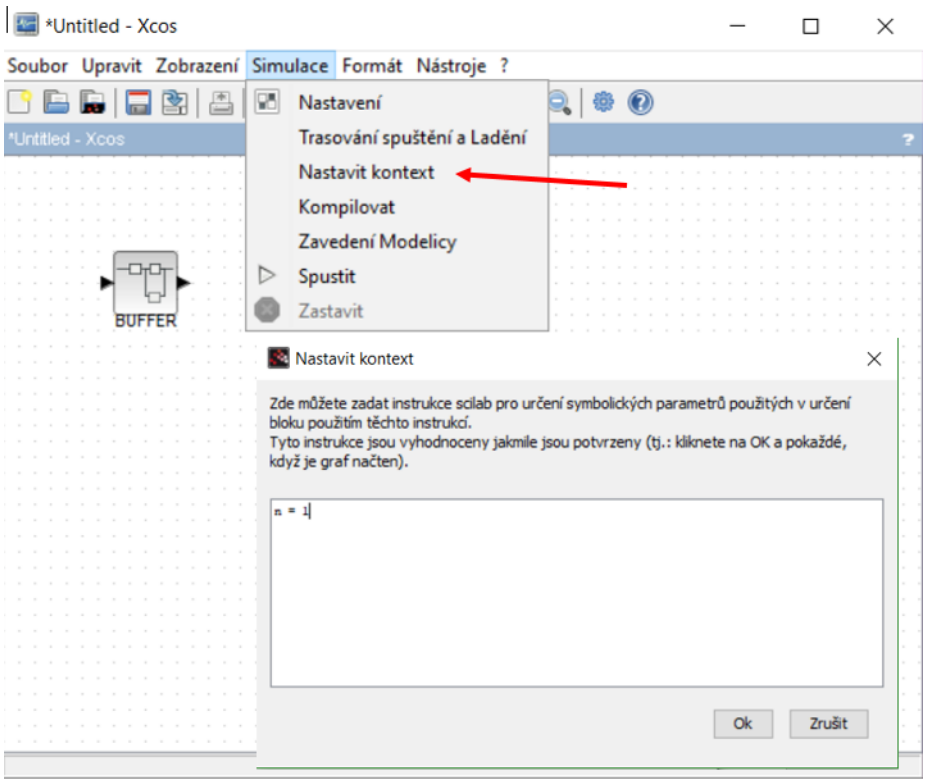
Vysílají se poslední hodnoty "n"vstupu na výstupní vektor velikosti "n".

void **outptr - ukazatel na běžné výstupní porty
void **inptr - ukazatel na běžné vstupní porty
int *outsz - velikost běžného vstupního portu

block->outsz - typu long
block->outptr - typu void
block->inptr - typu void

Po všech těchto krocích klikneme na tlačítko **ok**. Nyní se nám zobrazí okno pro zadání externích knihoven. Zde však nemusíme nic zadávat a můžeme pokračovat stisknutím tlačítka **ok**. Výpočetní funkce bloku je tímto hotová. [1]

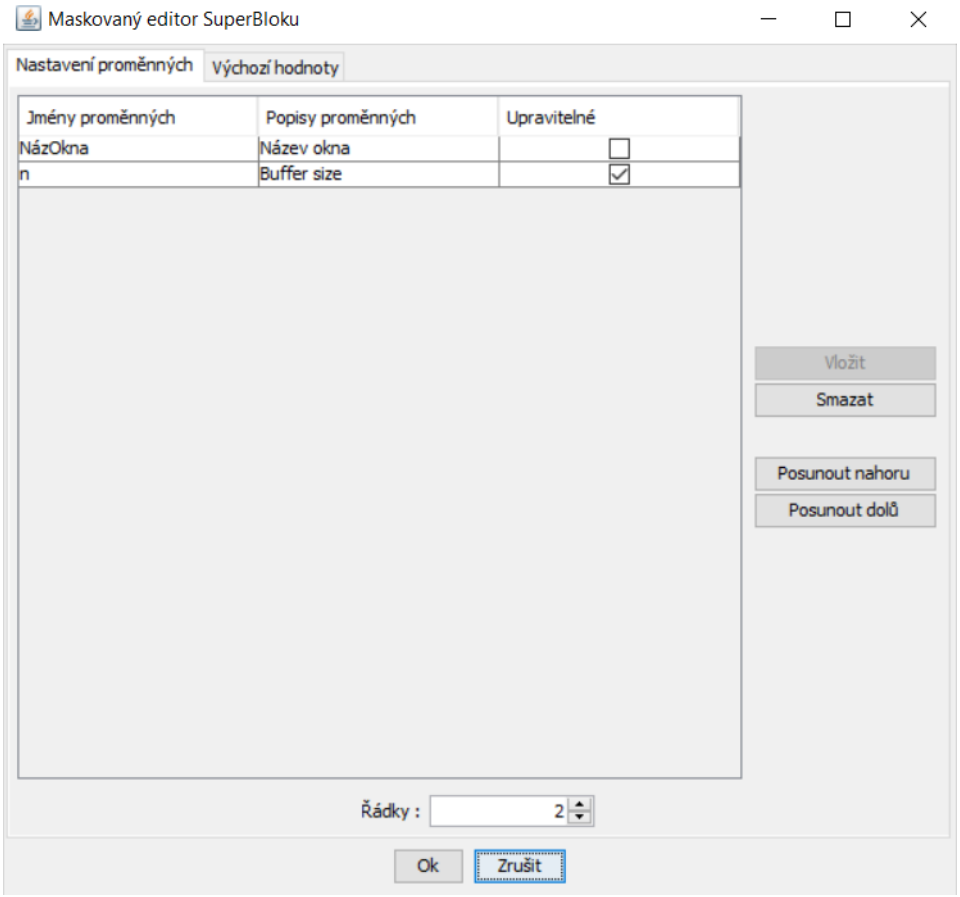
My si ale blok následně převedeme na *superblok* a také mu vytvoříme masku [1], aby se uživatel nemusel neustále vším proklikávat, jak jsme to dělali doposud. Vezmeme si tedy náš blok a připojíme mu ke vstupu blok *IN_f*, k výstupu pak blok *OUT_f* z palety *Port & Subsystem palette* neboli *Porty a Podsystém* a v poslední řadě k událostnímu vstupu blok *CLKINV_f*. Dále označíme náš blok včetně spojů (bez nově připojených portů), klikneme na něj a zvolíme volbu selection to superblok. Nyní můžeme bloky *IN_f*, *OUT_f* a *CLKINV_f* smazat. Blok si také přejmenujeme na *BUFFER*, opět jen pro lepší přehlednost. Dále v nabídce Xcos klikneme na volbu **simulace** a zde na volbu **Nastavit kontext**. Objeví se nám nyní okno, do kterého zadáme proměnnou, například n a nastavíme jí prozatím hodnotu 1. Postup lze vidět na následujícím obrázku 31.



Obrázek 31: Nastavení kontextu

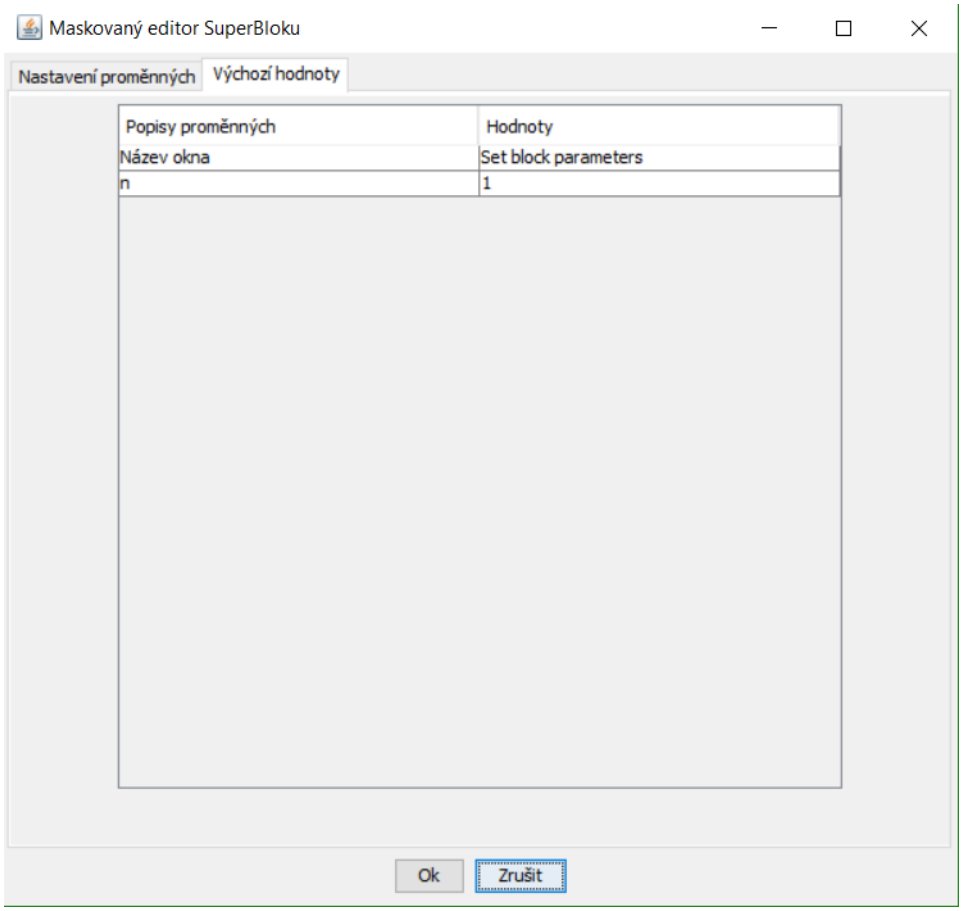
[1] U verze Scilabu 6.0.2 vytvoření masky nefungovalo, nedá se přiřadit kontext. Masku jsem tedy vytvářel ve verzi 5.5.2 u které jsem věděl, že to jde.

Nyní si rozklikneme náš superblok a otevřeme si vnitřní blok *CBLOCK*. V první tabulce do kolonky *output port sizes* vepíšeme naši proměnnou *n*. Potvrdíme. Dále se nám opět objeví okno pro pro definici C funkce. Zde nic neměníme a jen potvrdíme. Dále našemu superbloku přiřadíme masku. Klikneme tedy pravým tlačítkem myši na superblok, vybereme možnost **maska superbloku** a zde možnost **vytvořit**. Tento postup zopakujeme, jen místo **vytvořit** vybereme možnost **přizpůsobit...**. Nyní se nám otevře okno pro úpravu masky, jak lze vidět na následujícím obrázku 32.



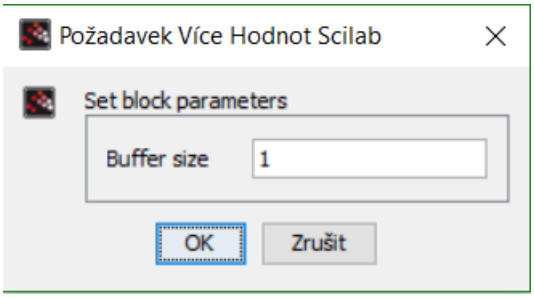
Obrázek 32: Editor masky

V tomto okně si přidáme náš kontext *n* a dáme mu název **Buffer size**. V dalším okně, které lze vidět na dalším obrázku 33, si můžeme nastavit libovolnou výchozí hodnotu.



Obrázek 33: Editor masky, nastavení výchozí hodnoty

Jakmile potvrdíme, můžeme kliknout na blok a uvidíme, že se nedostaneme dovnitř, ale objeví se nám okno s požadavkem pro zadání **Buffer size** a nastavená výchozí hodnota, jak lze vidět na následujícím obrázku 34.



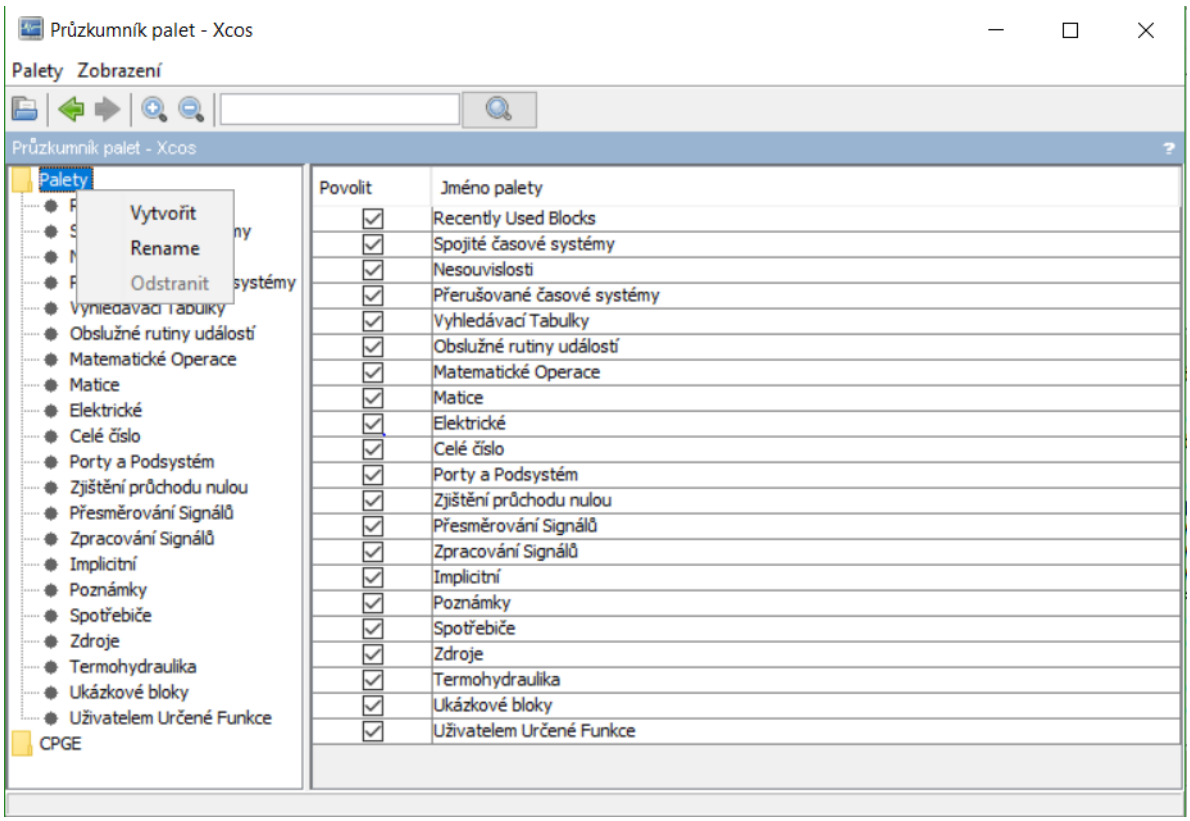
Obrázek 34: Nastavení hodnoty Buffer size

V této podkapitole jsem čerpal ze zdrojů [1, 12].

4.2 Začlenění bloku do uživatelské palety

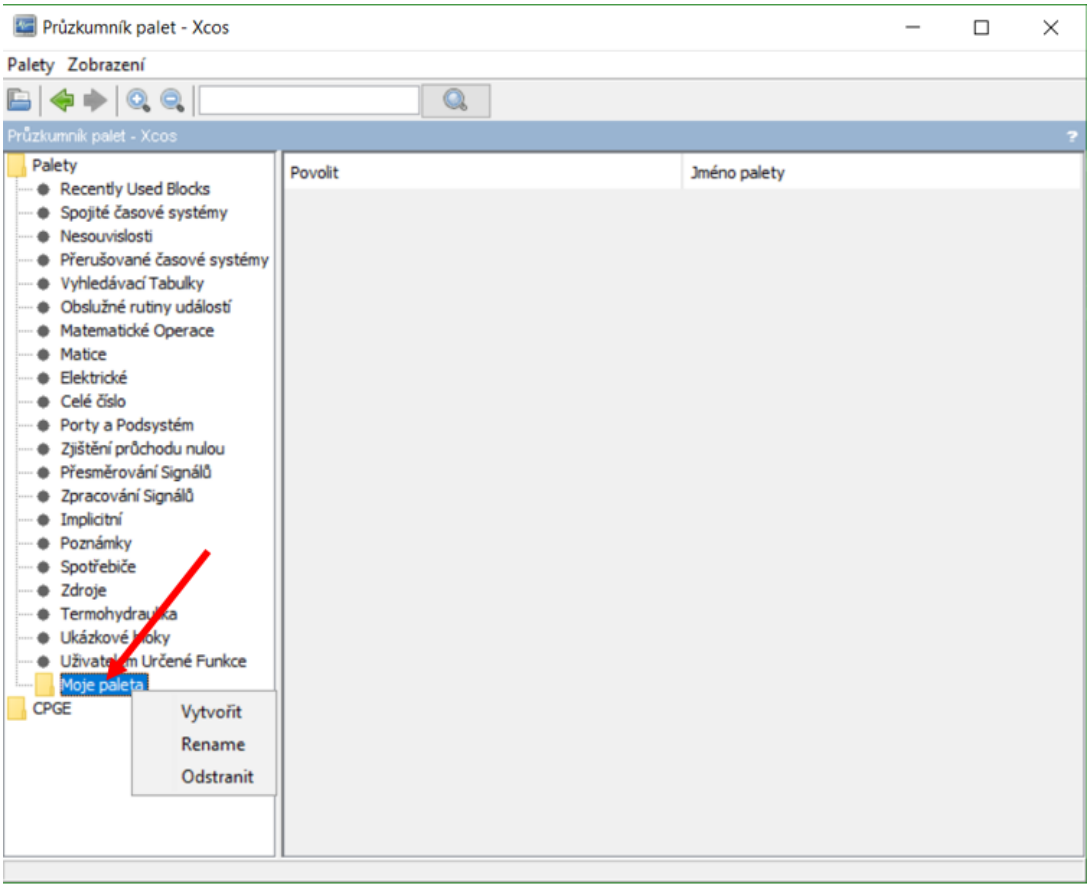
Nyní si tedy náš nově vytvořený blok začleníme do nově vytvořené palety, kterou si nazveme například Moje paleta.

V prvním kroku si tedy jednoduše vytvoříme novou paletu. To provedeme tak, že v průzkumníku palet klikneme pravým tlačítkem myši na složku **Palety** a zvolíme možnost **vytvořit**. Postup lze vidět na následujícím obrázku 35.



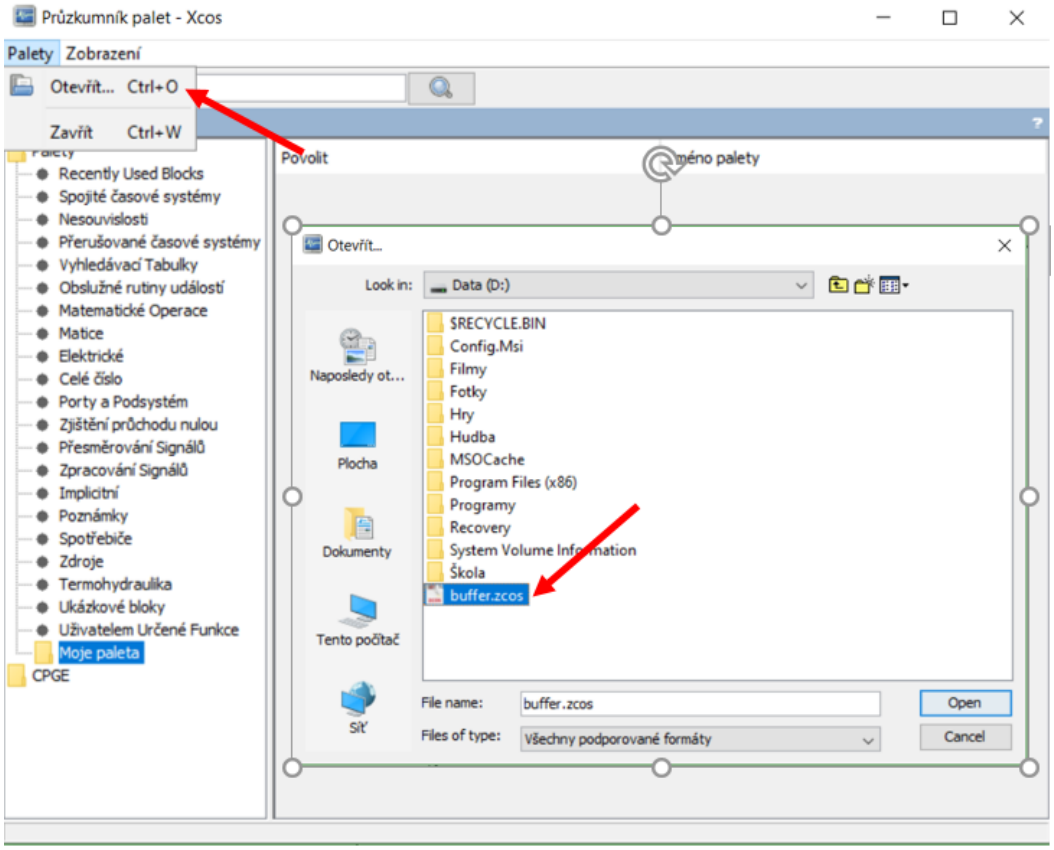
Obrázek 35: Vytvoření nové palety Moje paleta

Dále se nám objeví okno pro pojmenování palety. Nyní si můžeme všimnout, že se nám dole v průzkumníku palet vytvořila nová paleta, jak lze vidět na dalším obrázku 36. Pokud si budeme chtít paletu přejmenovat, stačí na ni kliknout pravým tlačítkem myši a zvolit volbu **rename**.



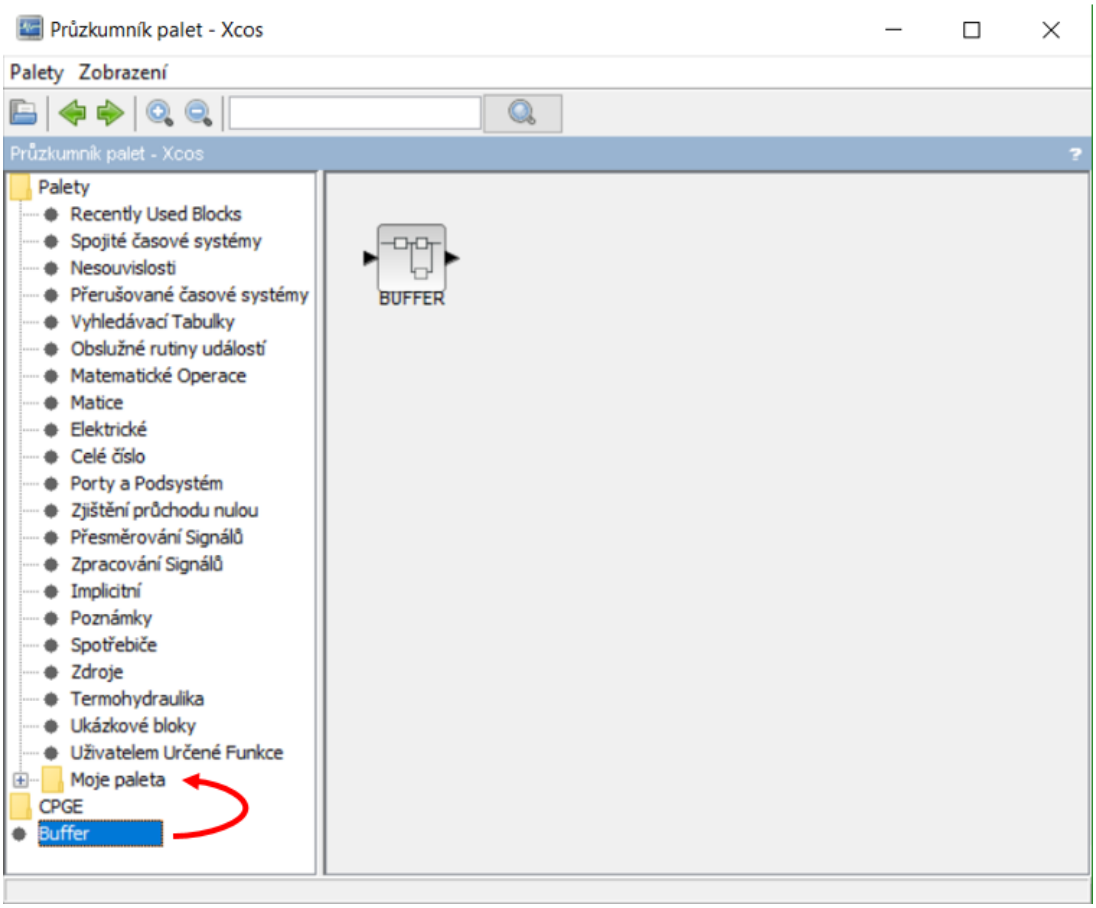
Obrázek 36: Modifikace nové palety

Nyní musíme do průzkumníku palet vložit náš blok. To provede tak, že v nabídce průzkumníku palet klikneme na nabídku **palety** a z této nabídky vybereme možnost **otevřít**. Najdeme si v počítači náš blok a ten vybereme. Postup lze vidět na následujícím obrázku 37.



Obrázek 37: Načtení bloku BUFFER do průzkumníku palet

Můžeme si všimnout, že se nám dole v průzkumníku palet přidal náš vybraný blok, jak můžeme vidět na následujícím obrázku 38. Musíme ho však pro lepší přehlednost přejmenovat. To docílíme tak, že na něj klikneme pravým tlačítkem myši a klikneme na volbu **rename**. Jakmile máme přejmenováno, musíme blok ještě vložit do naší palety. To uděláme tak, že jej jednoduše přetáhneme do naší palety *Moje paleta*.



Obrázek 38: Přesunutí bloku BUFFER do palety Moje paleta

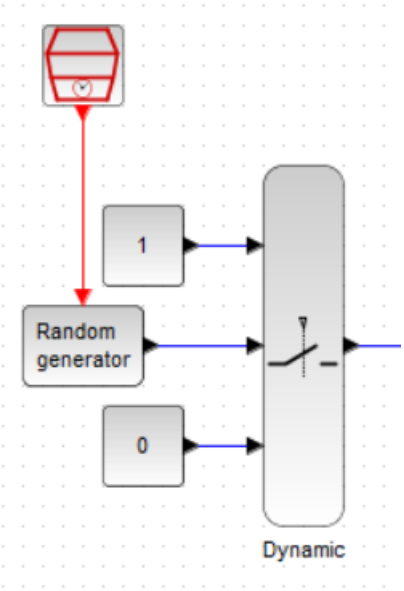
Po všech těchto krocích máme konečně blok *BUFFER* vsazený v paletě *Moje paleta* a můžeme si jej kdykoliv z této palety načíst do prostředí Xcos a pracovat s ním dle libosti. Tato paleta nám zde zůstane napořád i po restartování počítače.

V této podkapitole jsem čerpal ze zdrojů [13].

4.3 Příklad zapojení s blokem BUFFER

V této podkapitole si ukážeme příklad zapojení s nově vytvořeným blokem *BUFFER*.

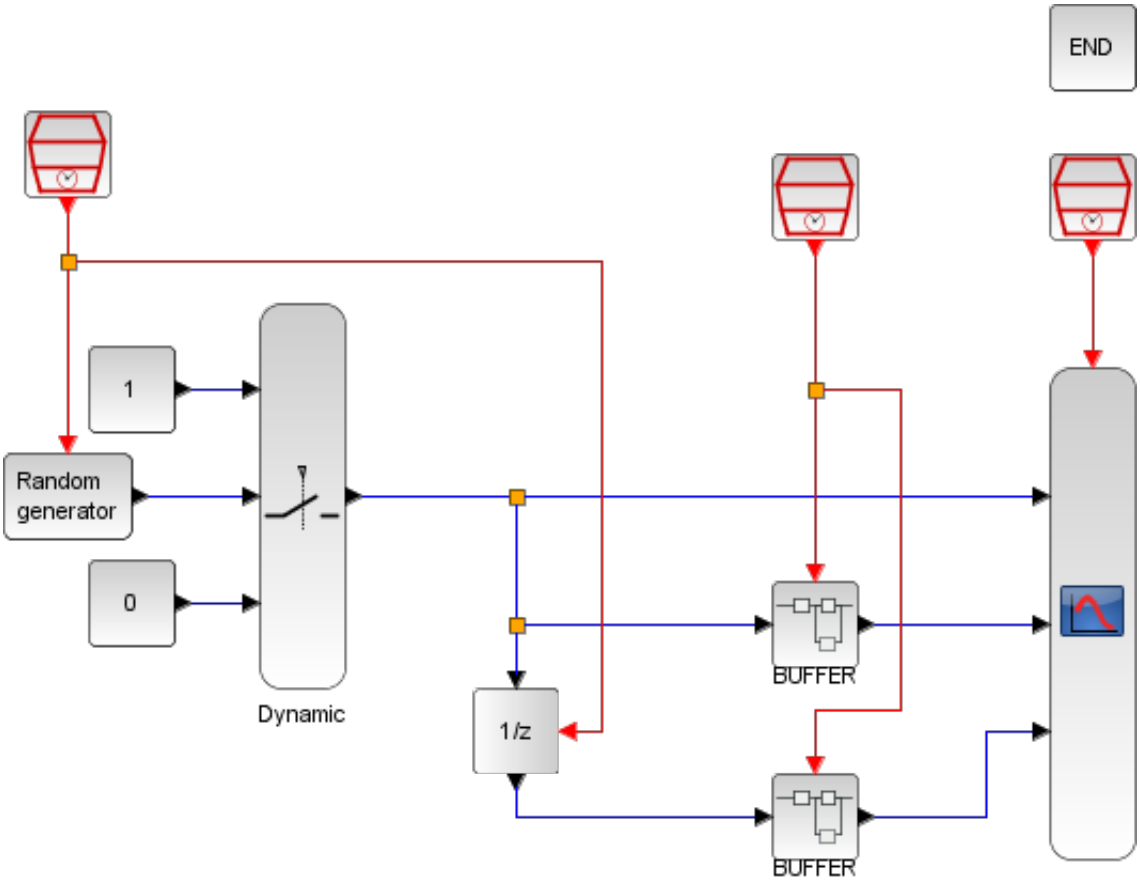
Nejdříve si sestrojíme náhodný generátor jedniček a nul. Pro toto zapojení budeme potřebovat náhodný generátor *RAND_m*, který můžeme najít v paletě *Zdroje* neboli *Sources palette*, dále konstanty 1 a 0 *CONST_m* z té samé palety, přepínač *SWITCH2_m* z palety *Přesměrování Signálů* neboli *Signal routing palette* a v poslední řadě časový vzorkovač *SampleCLK* opět z palety *Zdroje*. Konstanty 1 a 0 připojíme na první a třetí vstup přepínače a náhodný generátor na druhý vstup. Generátor tedy bude fungovat tak, že bude náhodně generovat hodnoty z prvního a třetího vstupu, čili naše konstanty 1 a 0. Zapojení lze vidět na následujícím obrázku 39.



Obrázek 39: Generátor 0 a 1

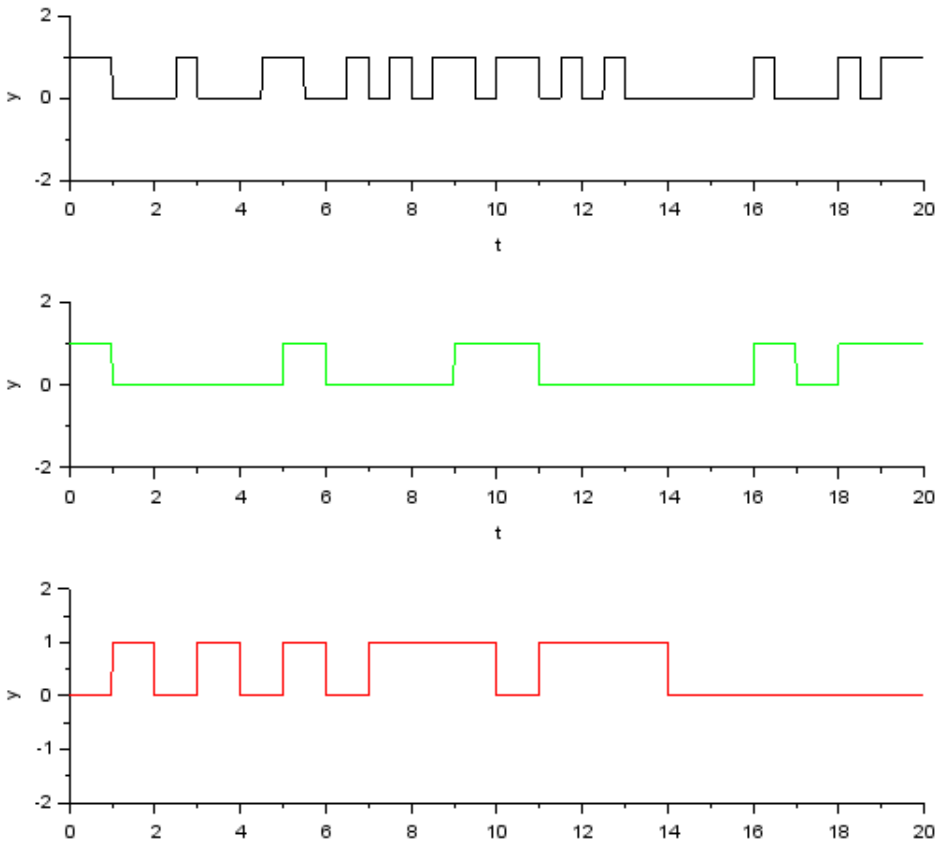
U bloku *SampleCLK* nastavíme Sample time na 1/2, u náhodného generátoru se nic měnit nebude (Datatype bude roven jedné, flag nule, A také nule, B jedné) a u přepínače změníme jen práh neboli threshold na 0.5.

Dále za náš generátor připojíme blok *DOLLAR_f*, který slouží jako zpožďovač signálu. Dále si před a za tento zpožďovač přípojíme náš blok *BUFFER*. K těmto blokům také připojíme časové vzorkovače *SampleCLK*. Zpožďovač můžeme připojit ke vzorkovači u generátoru, se vzorkovacím časem 1/2. Vzorkovači, který je připojen na naše dva bloky *BUFFER* nastavíme vzorkovací čas na 1. Za oba bloky *BUFFER* už připojíme jen vykreslovač signálů *CMSCOPE*, který ještě připojíme na vzorkovač se vzorkovacím časem 1/100. Toto zapojení můžeme vidět na dalším obrázku 40.



Obrázek 40: Příklad zapojení s blokem BUFFER

Po spuštění se nám vykreslí 3 signály. Jelikož všechny tři okruhy mají rozdílnou rychlost, můžeme tak vidět, jak se tyto signály v čase liší. Vykreslení signálů můžeme vidět na obrázku 41.



Obrázek 41: Vykreslení grafů zapojení

5 Možnosti sdílení uživatelsky vytvořených palet mezi jednotlivými uživateli

V této kapitole si ukážeme, jak lze sdílet palety vytvořené uživatelem. Připomeneme si také jak paletu vytvořit a jak do ní začlenit kterýkoliv blok.

V první řadě je tedy důležité si připomenout, jak vytvořit paletu a začlenit do ní blok. Všechny příkazy budeme psát do příkazového řádku SciNotes.
Nejdříve musíme načíst Xcos knihovny pomocí následujícího příkazu:

```
loadXcosLibs();
```

Výpis 10: Načtení Xcos knihoven

Dále si vytvoříme instanci nové palety a pojmenujeme si ji například *My palette* pomocí příkazu:

```
pal = xcosPal("My palette");
```

Výpis 11: Vytvoření instance nové palety

V dalším kroku si zvolíme, který blok chceme do palety vsadit. My si pro ukázku zvolíme již známý blok *BIGSOM_f*. Tomu přiřadíme název *o1*. Docílíme toho následujícím příkazem:

```
o1 = BIGSOM_f("define");
```

Výpis 12: Výběr bloku pro vsazení do nové palety

Nyní si náš vybraný blok začleníme do nové palety pomocí příkazu:

```
pal = xcosPalAddBlock(pal, o1);
```

Výpis 13: Přiřazení bloku do nové palety

A v poslední řadě si naši paletu exportujeme do souboru s názvem *My palette.sod*. Já si ji, jak lze vidět z následujícího výpisu, ukládám na disk D.

```
xcosPalExport(pal, "d:/My palette.sod");
```

Výpis 14: Export palety do souboru

Po všech těchto krocích se nám vytvořil soubor *My palette.sod* tam, kam jsme nastavili cestu pro vytvoření. Tento soubor můžeme předat dalšímu uživateli, buď přes nějaký software, např e-mail, nebo hardware, např USB flash disk. Uživatel si jej vloží do svého počítače a následně otevře ve Scilabu v příkazovém řádku pomocí následujícího příkazu:

```
xcosPalAdd('d:/My palette.sod');
```

Výpis 15: Načtení palety ze souboru

Nyní po otevření Xcosu bude mít uživatel novou paletu *My palette*, včetně bloků do ní vložených, vloženou do průzkumníku palet.

V této kapitole jsem čerpal ze zdroje [11].

Závěr

Hlavním cílem této práce bylo popsat možnosti tvorby nových bloků pro systém Scilab a jeho rozšiřující balíček Xcos. Text v této práci obsahuje také mnoho obrázků od názorných ukázek prostředí pro lepší orientaci, přes ukázky různých bloků a dialogových oken až po jednoduchá schémata, aby čtenář řešené příklady co nejlépe pochopil. Také byl do přílohy umístěn vytvořený blok i všechny ukázky schémat, se kterými lze dle potřeb dále pracovat.
Výhoda celého systému Scilab je v tom, že je open-source, takže jej ocení lidé, kteří nechtějí utrácet za licence drahých softwarů, se kterými je Scilab podobný, na druhou stranu nevýhoda zde je taková, že tento systém není tak úplně propracovaný a občas se dá narazit na pár chyb, jak se mi také podařilo ve třetí kapitole při vytváření superbloku. Zde se mi nedařilo vytvořený superblok zprovoznit, protože se uvnitř superbloku stávalo, že se spoje mezi jednotlivými bloky samy duplikují. Na další problém jsem narazil při vytváření masky superbloku, kdy se do masky nechtěly načíst proměnné z kontextu. Tento problém jsem musel řešit instalací starší verze Scilabu a následné realizace masky v této starší verzi.
Při vytváření bloku BUFFER jsem zvolil postup přes takzvaný blok CBLOK, jelikož umožňoval jednoduchý postup při realizaci a také se při práci dalo hezky pracovat s literaturou uvedenou v seznamu literatury [1]. Po vytvoření bloku jsem se následně pokusil blok přidat do již existující palety s názvem Uživatelem Určené Funkce, kterou Xcos standardně nabízí, jenomže přidávání uživatelem vytvořených bloků do standardních výchozích palet není možné. U všech ukázek, u kterých jsem libovolné bloky přidával do palet, jsem musel nejdřív vytvořit palety nové a až následně do nich bloky vkládat. Při vložení nově vytvořeného bloku BUFFER do palety jsem nevolil postup přes příkazový řádek ve Scilabu, ale postupoval jsem ručně v Xcosu z toho důvodu, že při takovémto postupu se paleta s blokem po restartování programu Scilab nevytratí a zůstane zde do doby, dokud ji sám uživatel nevymaže.
Tento manuál může být přínosný těm uživatelům, kteří se chtějí naučit jak základní, tak pokročilejší práci s bloky v systému Xcos a také hlavně pokud chtějí nějaké nové bloky vytvářet.

Literatura

[1] CAMPBELL, Stephen J, Jean-Philippe CHANCELIER a Ramine NIKOUKHAH. Modeling and simulation in Scilab/Scicos with Scicoslab 4.4. Second edition. New York: Springer, [2010]. ISBN 978-1-4419-5526-5.

[2] Scilab - Open source software for numerical computation: About Scilab [online]. [cit. 2018-11-08]. Dostupné z: https://www.scilab.org/en/scilab/about

[3] Scilab. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2018 [cit. 2018-11-08]. Dostupné z: https://cs.wikipedia.org/wiki/Scilab;

[4] ČERNÝ, Michal. Scilab: simulace i výpočty s toolboxy. In: Root.cz [online]. Internet Info, 25.5.2010 [cit. 2018-11-19]. Dostupné z: https://www.root.cz/clanky/scilab-simulace-i-vypocty-s-toolboxy/

[5] Scicos_graphics. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-25]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/scicos_graphics.html

[6] Scicos_model. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-23]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/scicos_model.html

[7] Save. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-07]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/save.html

[8] XcosPalAddBlock. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-07]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/xcosPalAddBlock.html

[9] XcosPalAdd. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-07]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/xcosPalAdd.html

[10] SCHMIDT, Phil. Tutorial, Creating a C Function Block in Scicos. Scicos [online]. INRIA, c1994-2015, 7. březen, 2009 [cit. 2019-04-23]. Dostupné z: http://www.scicos.org/ScicosCBlockTutorial.pdf

[11] XcosPalExport. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-08]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/xcosPalExport.html

[12] Scicos Team. Constructing new blocks in Scicos. Scicos [online]. INRIA, c1994-2015, 3. březen, 2009 [cit. 2019-04-23]. Dostupné z: http://www.scicos.org/Formation_scicos_mars_2008.pdf

[13] Customizing Xcos with new Blocks and Palettes. Openeering [online]. Trento, Itálie: Enginsoft, 2012 [cit. 2019-04-20]. Dostupné z: http://www.openeering.com/sites/default/files/LHY_Customizing_Xcos_Block_Palette.pdf

[14] Buffer. ITBiz [online]. Nitemedia [cit. 2019-04-20]. Dostupné z: https://www.itbiz.cz/slovnik/informacni-technologie-it/buffer

[15] Scifunc_block_m. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-20]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/scifunc_block_m.html

[16] SUPER_f. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-20]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/SUPER_f.html

[17] BIGSOM_f. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-20]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/BIGSOM_f.html

[18] CBLOCK. Scilab Online Help [online]. Scilab Enterprises, c2011-2017 [cit. 2019-04-20]. Dostupné z: https://help.scilab.org/docs/6.0.2/en_US/CBLOCK.html

Seznam příloh

BIGSOM_f_puvodni-example.zcos	Schéma s neupraveným blokem BIGSOM_f
BIGSOM_f_example.zcos	Schéma s upraveným blokem BIGSOM_f
BIGSOM_f_example.sci	Funkce rozhraní upraveného bloku BIGSOM_f
BIGSOM_f_example.bin	Binární soubor upraveného bloku BIGSOM_f
buffer.zcos	Vytvořený blok BUFFER
buffer-example.zcos	Schéma s vytvořeným blokem BUFFER
scifunf-example.zcos	Schéma s blokem scifunc_block_m
superblok-example.zcos	Schéma s blokem SUPER_f
My palette.sod	Paleta My palette exportovaná do souboru